Corso sul linguaggio C++

Modulo 3

3 – Libreria conio.h

Prerequisiti

- Programmazione base
- Utilizzo di funzioni

Introduzione

In questa lezione fissiamo l'attenzione su come gestire l''**I/O** (Input/Output) sullo schermo.

La libreria **conio.h** contiene numerose funzioni di utilità che operano sull'**interfaccia testuale** di I/O.

Le più comuni funzioni di questa libreria, consentono di:

- eseguire letture e stampe guidate
- impostare il colore del testo
- impostare il colore dello sfondo dello schermo
- cancellare il contenuto dello schermo

Argomenti

- Posizione del cursore
- Utilizzo dei colori
- Ciclo di ritardo
- Stampa del codice ASCII
- Dimensioni del cursore
- Funzioni di input
- Attributi del testo
- Caratteri grafici
- Finestre grafiche
- Riempimento e colori

Informazioni generali

Per poter realizzare applicazioni C++ che utilizzino le funzionalità della libreria **conio.h** è necessario:

- conoscere l'utilizzo delle funzioni in C++
- aver installato la libreria **conio.h** sul sistema e il relativo file (libconio.a) come descritto in *Dev C++ - Note per l'utilizzo*

Posizione del cursore

Programma *gotoxy.cpp* che stampa una scritta scorrevole da sinistra a destra, posizionata sulla riga 10.

```
#include <iostream>
#include <conio.h>
using namespace std;
int main()
{ int a,b,c;
  gotoxy (10, 10);   cout<<"Primo valore..: ";   gotoxy (28, 10);   cin>>a;
  gotoxy (10, 14);   cout<<"Secondo valore: " ;   gotoxy (28, 14); cin>>b;
  c=a+b;
  gotoxy (10, 18); cout<<"La somma e'...: "; gotoxy (28, 18);   cout<<c;
  gotoxy (40, 20);
  system ("Pause");
  return 0;
}
```

Posiziona il cursore sullo schermo

Posizione del cursore

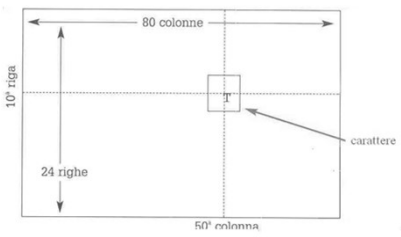
Posizione del cursore

In modalità testuale, lo schermo prevede normalmente, 25 righe ed 80 colonne.

La funzione

void gotoxy (**int** *colonna,* **int** *riga*)

consente il posizionamento del cursore in una qualunque delle 25 * 80 posizioni possibili.



Utilizzo dei colori

Programma *ScrittaScorrevole.cpp*

```
#include <iostream>
#include <conio.h>
using namespace std;
int main()
{ string s;
  int riga=10, colonna=0;
  s="Scritta scorrevole";
  textbackground (7);  clrscr ();
  for (colonna=0;colonna<80; colonna++)
  {   textcolor (colonna%15);
      gotoxy (colonna, riga);    cout<<s;
      delay (150);
      clrscr ();
  }
  system ("Pause");
  return 0;
}
```

Colore dello sfondo

Pulizia dello schermo

Colore del testo

Ritardo

Utilizzo dei colori

Utilizzo dei colori

Impostazione colore testo

void textcolor (**int** c);
void textcolor (**COLORS** c);

Impostazione colore sfondo

void textbackground (**int** c);
void textbackground (**COLORS** c);

Dopo **textbackground()** occorre eseguire **clrscr()** per ricolorare lo schermo

Utilizzo dei colori

```
typedef enum
{
   BLACK,        BLUE,          GREEN,
   CYAN,         RED,           MAGENTA,
   BROWN,        LIGHTGRAY,     DARKGRAY,
   LIGHTBLUE,    LIGHTGREEN,    LIGHTCYAN,
   LIGHTRED,     LIGHTMAGENTA,  YELLOW,
   WHITE
} COLORS;
```

Ciclo di ritardo

Durante l'esecuzione può essere necessario applicare un ciclo di ritardo.
Il ritardo è realizzato mediante la funzione:

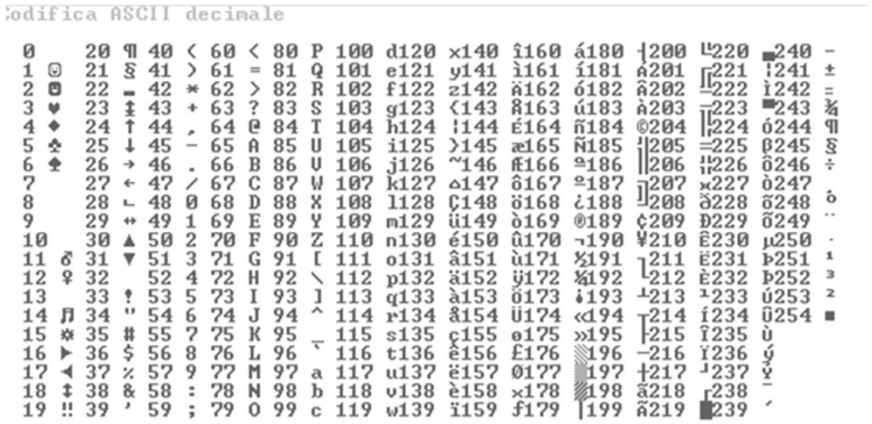
void delay (**unsigned** *msec*)

Stampa del codice ASCII

```
.....
int main()
{   int cod,riga=3, colonna=3;
    char c;
    textbackground (15);     clrscr ();
    cout<<"Codifica ASCII decimale\n";
    textcolor (3);
    for (cod=0; cod<255; cod++)
    {  gotoxy (colonna,riga);
       c=cod;
       cout<<cod<<" "<<c;  riga++;
       if (riga > 22)  {  riga = 3;
                          colonna=colonna + 5;    }
    }
    gotoxy (10,24);
    system ("Pause");
   return 0;
↓
```

Il programma *Ascii.cpp* stampa la serie dei caratteri ASCII con il loro codice **decimale**, sotto forma di tabella su sfondo bianco.

Stampa del codice ASCII



Stampa del codice ASCII

```
.....
int main()
{   int cod,riga=3, colonna=3;
   char c;
   textbackground (15);     clrscr ();
   cout<<"Codifica ASCII esadecimale\n";
   textcolor (5);
   for (cod=0; cod<255; cod++)
   {  gotoxy (colonna,riga);  c=cod;
      cout.flags(ios::hex);
      cout<<cod<<" "<<c;  riga++;
      if (riga > 22)  {  riga = 3;
                         colonna=colonna + 5;     }
   }
   gotoxy (10,24);  system ("Pause");
  return 0;
}
```

Il programma stampa la serie dei caratteri ASCII con il loro codice **esadecimale**, sotto forma di tabella su sfondo bianco.

Stampa del codice ASCII

Codifica ASCII esadecimale

0		14	¶	28	(	3c	<	50	P	64	d	78	x	8c	î	a0	á	b4	┤	c8	└	dc	▄	f0	≡
1	☺	15	§	29	)	3d	=	51	Q	65	e	79	y	8d	ì	a1	í	b5	Á	c9	╔	dd	¦	f1	±
2	☻	16	▬	2a	*	3e	>	52	R	66	f	7a	z	8e	Ä	a2	ó	b6	Â	ca	╩	de	ì	f2	‗
3	♥	17	↨	2b	+	3f	?	53	S	67	g	7b	{	8f	Å	a3	ú	b7	À	cb	╦	df	▀	f3	¾
4	♦	18	↑	2c	,	40	@	54	T	68	h	7c	¦	90	É	a4	ñ	b8	©	cc	╠	e0	ó	f4	¶
5	♣	19	↓	2d	-	41	A	55	U	69	i	7d	}	91	æ	a5	Ñ	b9	╣	cd	═	e1	ß	f5	§
6	♠	1a	→	2e	.	42	B	56	V	6a	j	7e	~	92	Æ	a6	ª	ba	║	ce	╬	e2	Ô	f6	÷
7		1b	←	2f	/	43	C	57	W	6b	k	7f	⌂	93	ô	a7	º	bb	╗	cf	¤	e3	Ò	f7	¸
8		1c	∟	30	0	44	D	58	X	6c	l	80	Ç	94	ö	a8	¿	bc	╝	d0	ð	e4	õ	f8	°
9		1d	↔	31	1	45	E	59	Y	6d	m	81	ü	95	ò	a9	®	bd	¢	d1	Ð	e5	Õ	f9	¨
a		1e	▲	32	2	46	F	5a	Z	6e	n	82	é	96	û	aa	¬	be	¥	d2	Ê	e6	µ	fa	·
b	♂	1f	▼	33	3	47	G	5b	[	6f	o	83	â	97	ù	ab	½	bf	┐	d3	Ë	e7	Þ	fb	¹
c	♀	20		34	4	48	H	5c	\	70	p	84	ä	98	ÿ	ac	¼	c0	└	d4	È	e8	Þ	fc	³
d		21	!	35	5	49	I	5d	]	71	q	85	à	99	Ö	ad	¡	c1	┴	d5	ı	e9	Ú	fd	²
e	♫	22	"	36	6	4a	J	5e	^	72	r	86	å	9a	Ü	ae	«	c2	┬	d6	Í	ea	Û	fe	■
f	☼	23	#	37	7	4b	K	5f	_	73	s	87	ç	9b	ø	af	»	c3	├	d7	Î	eb	Ù		
10	►	24	$	38	8	4c	L	60	`	74	t	88	ê	9c	£	b0	▒	c4	─	d8	Ï	ec	ý		
11	◄	25	%	39	9	4d	M	61	a	75	u	89	ë	9d	Ø	b1	▒	c5	┼	d9	┘	ed	Ý		
12	↕	26	&	3a	:	4e	N	62	b	76	v	8a	è	9e	×	b2	▒	c6	ã	da	┌	ee	¯		
13	‼	27	'	3b	;	4f	O	63	c	77	w	8b	ï	9f	ƒ	b3	│	c7	Ã	db	█	ef	´		

Dimensioni del cursore

Il programma *CurDimension.cpp* è in grado di modificare l'aspetto del cursore

```
#include <iostream>
#include <conio.h>
using namespace std;
int main()
{ string s;
  cout<<"cursore normale";
  _setcursortype (_NORMALCURSOR); cin>>s;
  cout<<"cursore invisibile";
  _setcursortype (_NOCURSOR); cin>>s;
  cout<<"cursore sottile";
  _setcursortype (1);
  cin>>s;
  cout<<"cursore a blocco";
  _setcursortype (_SOLIDCURSOR); cin>>s;
  system("Pause");
  return 0;
}
```

Dimensioni del cursore

Codici numerici equivalenti

void _setcursortype (int n)

_LNORMALCURSOR ←→ 25
_NOCURSOR ←→ 0
_SOLIDCURSOR ←→ 100
Sottile ←→ 1

Funzioni di input

Lettura senza echo

int getch ();

Lettura con echo

int getche ();

Pressione di un qualunque tasto

int kbhit ();

Funzioni di input

```
#include <iostream>
#include <conio.h>
using namespace std;
int main()
{  cout<<"getch() - lettura tasto senza echo"<<endl;
   getch();
   cout<<"getche() - lettura tasto con echo"<<endl;
   getche();
   cout<<"kbhit() - pressione qualunque tasto"<<endl;
   while (!kbhit())
         cout<<"attendo!"<<endl;
   system("Pause");
   return EXIT_SUCCESS;
}
```


Attributi del testo

```
#include <iostream>
#include <conio.h>
using namespace std;
int main()
{  text_info r;
   gettextinfo(&r);
   cout<<(int)r.winleft<<endl;          // ascissa minima finestra
   cout<<(int)r.wintop<<endl;           // ordinata minima finestra
   cout<<(int)r.winright<<endl;         // ascissa massima finestra
   cout<<(int)r.winbottom<<endl;        // ordinata massima finestra
   cbut<<(int)r.attribute<<endl;        // colore corrente sfondo
   cout<<(int)r.normattr<<endl;         // colore corrente testo
   cout<<(int)r.currmode<<endl;         // modo corrente
   cout<<(int)r.screenheight<<endl;     // altezza schermata
   cout<<(int)r.screenwidth<<endl;      // larghezza schermata
   cout<<(int)r.curx<<endl;             // ascissa corrente
   cout<<(int)r.cury<<endl;             // ordinata corrente
   system("Pause");
   return EXIT_SUCCESS;
}
```

Attributi del testo

La funzione usata è:

void gettextinfo (**text_info** *r);

dove text_info è definito come:

```
typedef struct
{
   unsigned char winleft, wintop, winright, winbottom,
    attribute, normattr, currmode, screenheight, screenwidth, curx, cury;
} text_info;
```


Caratteri grafici

Mediante l'istruzione

cout.flags(ios::**hex**);

è possibile stampare i **caratteri grafici** tramite il loro codice esadecimale.

```
void hline (int x, int y, int len, COLORS c)
{  int i;
   int curcolor;
   text_info r;
   gettextinfo (&r);                  /* preleva caratteristiche del testo */
   curcolor=(int) r.normattr;         /* salva colore corrente */
   gotoxy (x,y);
   textcolor (c);                     /* disegna linea in colore */
   for (i=1; i<=len; i++)
     cout<<"\x0c4";                   /* codice hex di "-"
   textcolor (curcolor);              /* ripristina colore di partenza */
}
```

Caratteri grafici

Il **main()** ha il seguente aspetto:

```
#include <iostream>
#include <conio.h>
void hline (int x, int y, int len, COLORS c);
using namespace std;
int main()
{
   hline (10, 10, 20, GREEN);     /* istanza della funzione hline() */
   cout<<endl;
   system("Pause");
   return 0;
}
```


Finestre grafiche

```
#include <iostream>
#include <conio.h>
using namespace std;
int main()
{ int larghezza=0, altezza=0;
  con_setwindow (0, 0, 5, 5);
  do
  {  larghezza+=2; altezza++;    delay (2);
     con_resize (larghezza, altezza);
  } while (altezza<55 && larghezza<50);
  textbackground (LIGHTBLUE);     clrscr();
  con_settitle ("Finestra di prova");
  gotoxy (5, 5);   cout<<"ciao";
  gotoxy (5, 10);   system("Pause");
  return 0;
}
```

Mediante apposite funzioni grafiche è possibile l'esecuzione di un programma all'interno di una **finestra grafica**.

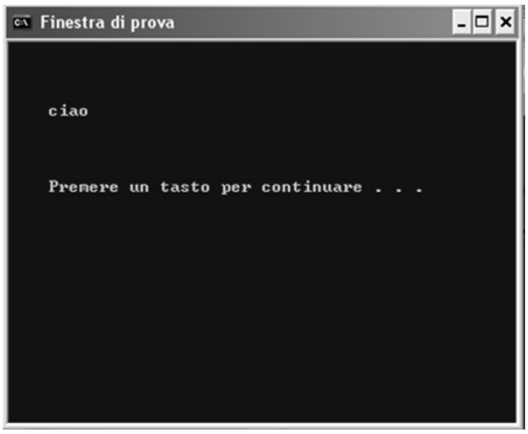
Creazione di una **finestra grafica**.

Ridimensionamento di una **finestra grafica**.

Impostazione del titolo di una **finestra grafica**.

Finestre grafiche

L'esecuzione del programma avviene all'interno di una finestra grafica



Finestre grafiche

Le funzioni di utilità più frequenti sono:

void con_setwindow (**int** left, **int** top, **int** right, **int** bottom);

void con_resize (**int** width, **int** height);

void con_settitle (const char *str);

Riempimento e colori

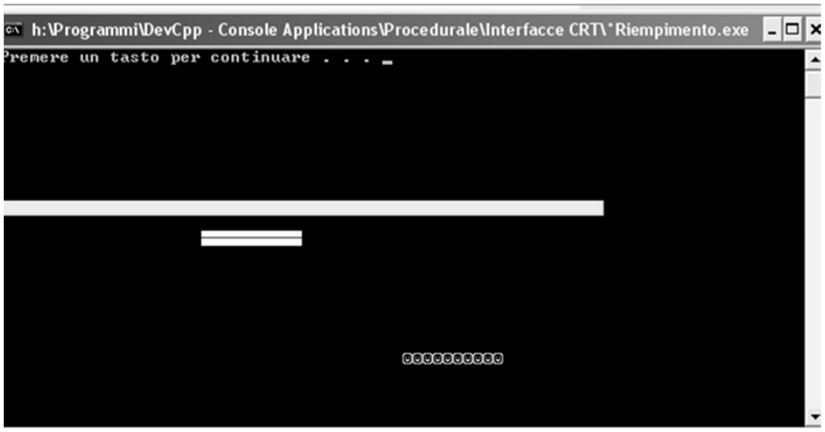
```
#include <iostream>
#include <conio.h>
using namespace std;
int main()
{ int riga=10, colonna=0;
  do
  {   con_fillattr (5, 14, 1, colonna, riga);
      colonna++;     delay (50);
   } while (colonna<60);
  con_fill ('\xc4', 12, 15, 10, 20, 12);
  con_fillcr ('\x001', 10, 40, 20);
  system ("Pause");
  return 0;
}
```

Mediante apposite funzioni grafiche è possibile riempire zone di schermo con *colori, posizione ed estensione variabili*

Riempimento con colore di testo 5 e sfondo 14, per simulare una *progress bar.*

Riempimento di un'area con un dato carattere

Riempimento e colori



Riempimento e colori

Le funzioni per ottenere effetti di riempimento più usate, sono:

void con_fillattr (**int** fg, **int** bg, **int** n, **int** x, **int** y);
void con_fill (**char** c, **int** fg, **int** bg, **int** n, **int** x, **int** y);
void con_fillcr (**char** c, **int** n, **int** x, **int** y);

Cosa ho imparato

- Le funzioni di schermo
- Caratteristiche di un'interfaccia utente

Cosa ho imparato a fare

- Utilizzare la libreria **conio.h**
- Progettare e realizzare interfacce utente

Terminologia

- Libreria **conio.h**
- **gotoxy** ()
- **textcolor** ()
- **textbackground** ()
- **delay** ()
- **clrscr** ()
- **_setcursortype** ()
- **gettextinfo** ()
- **con_setwindow** ();
- **void con_resize** ();
- **void con_settitle** ();
- **void con_fillattr** ();
- **void con_fill** ();
- **void con_fillcr** ();

Altre fonti di informazione

- www.delorie.com/djgpp/doc/libc/libc_4.html