Osnove

Za naprednejše upravljanje s terminalom v C++ programu uporabljamo knjižnico `ncurses`. Enostaven program izgleda tako:

```
#include <ncurses.h>

int main() {
    // Inicializiramo in nastavimo knjižnico
    initscr();
    cbreak();
    noecho();
    keypad(stdscr, true);

    printw("Hello World!"); // Izpišemo besedilo
    refresh(); // Posodobimo zaslon

    getch(); // Pokačamo, da uporabnik pritisne tipko
    endwin(); // Zaključimo s programom
    return 0;
}
```

Funkcije `initscr()`, `cbreak()`, `noecho()` in `keypad()` moramo vedno poklicati na začetku programa, da bo knjižnica `ncurses` pravilno delovala. Za izpisovanje na zaslon lahko uporabimo funkcijo `printw()`, ki deluje enako kot že znani `printf()`, le da izpisuje na pravo mesto. Včasih se izpisano besedilo ne bo takoj pojavilo na zaslonu. Da se bo sigurno izpisalo, moramo poklicati funkcijo `refresh()`.
Če želimo počakati, da bo uporabnik pritisnil tipko na tipkovnici, pokličemo funkcijo `getch()`. Ta funkcija nam tudi vrne tipko, ki jo je uporabnik pritisnil, ki pa jo tu zavržemo.
Na koncu programa moramo vedno poklicati funkcijo `endwin()`, da se terminal vrne v prejšnje stanje.

Ko prevajamo kodo, ki uporablja knjižnico `ncurses`, moramo k ukazu za prevajanje dodati še argument `-lncurses` (prvi znak je mali L). V urejevalniku Geany pravo nastavitev najdemo pod *Build > Set Build Commands*, nato pa na konec ukaza pri *Compile* dodamo še `-lncurses`.

Pisanje na zaslon

Besedilo pišemo na zaslon s funkcijo `printw()`, ki deluje enako kot že znana funkcija `printf()`. Kot prvi argument ji podamo besedilo v dvojnih narekovajih, ki lahko vsebuje formatnike, v naslednjih argumentih pa podamo vsebino teh formatnikov. Besedilo se bo izpisalo na mesto, kjer stoji *kurzor*.
Kurzor lahko premikamo s funkcijo `move(y,x)`, ki ji podamo dve števili. Prvo število je *zaporedna vrstica* v terminalu, drugo pa *zaporedni stolpec*, kamor naj se kurzor premakne. Na primer, da izpišemo tri vrstice besedila eno pod drugo:

```
move(3, 10);
printw("Prva vrstica");
move(4, 10);
printw("Druga vrstica");
move(5, 10);
printw("Tretja vrstica");
```

To besedilo se bo izpisalo v tretjo, četrto in peto vrstico terminala, vse vrstice se bodo začele na desetem stolpcu. Funkciji `move()` in `printw()` pogosto kličemo eno za drugo, zato nam je na voljo bližnjica; zgornje besedilo bi lahko izpisali tudi z

```
mvprintw(3, 10, "Prva vrstica");
mvprintw(4, 10, "Prva vrstica");
mvprintw(5, 10, "Prva vrstica");
```

Funkcija `mvprintw(y,x,besedilo)` prvo premakne kurzor na mesto `(y,x)`, nato pa izpisala dano besedilo. Tudi ta funkcija lahko uporablja formatnike.

Branje podatkov

Ko pokličemo funkcijo `getch()`, bo program počakal, da uporabnik pritisne tipko, nato pa vrne kodo pritisnjene tipke. Če je tipka črka ali številka, dobimo njeno ASCII kodo, sicer pa dobimo posebno številko:

```
int crka = getch();
if (crka == 'a') {
    printw("Pritisnjen 'a'");
} else if (crka == '3') {
    printw("Pritisnjen '3'");
} else if (crka == KEY_UP) {
    printw("Puscica gor");
} else if (crka == '\n') {
    printw("Pritisnjen Enter");
}
```

Če želimo prebrati celotno vrstico, uporabimo funkcijo `getstr()` — ta počaka, da uporabnik vnese besedilo in stisne tipko Enter.

Atributi

Besedilu lahko dodamo oziroma odvzamemo atribute s funkcijama `attron()` in `attroff()`. Na primer, če želimo zapisati krepko besedilo:

```
attron(A_BOLD);
printw("Krepko besedilo ");
attroff(A_BOLD);
printw("Obicajno besedilo");
```

Najbolj uporabni atributi so `A_BLINK`, `A_BOLD`, `A_UNDERLINE` in `A_REVERSE`.

Barve

Če želimo v programu uporabljati različne barve, moramo poleg omenjenih funkcij na začetku programa poklicati še `start_color()`. Za osnovo uporabo moramo definirati par barv (barva besedila in barva ozadja) s funkcijo `init_pair()`:

```
start_color();
// ...

// Prva številka je poljubna, si jo pa zapomnimo
init_pair(42, COLOR_RED, COLOR_WHITE);

// Ko želimo uporabiti barvo, uporabimo to številko
// kot atribut.
attron(COLOR_PAIR(42));
printw("To je obarvano besedilo");
attroff(COLOR_PAIR(42));
```

Kot prvi argument funkciji `init_pair()` damo poljubno številko do 65 535, ki jo bomo uporabili kasneje, da par barv dejansko aktiviramo. To naredimo s funkcijo `attron()`, v katero zapišemo `COLOR_PAIR(x)` za našo izbrano številko `x`.
Barva se obnaša kakor vsi drugi atributi. Ko je ne želimo več uporabljati, jo lahko izklopimo z `attroff()`.

V knjižnici nam je na voljo osem barv (glej spodaj), lahko pa definiramo tudi svoje barve s funkcijo `init_color()`. Tej funkciji podamo štiri argumente: Prvi je številka med 8 in 255, ki jo bomo kasneje uporabili kot drugi oziroma tretji argument v `init_pair()`, ostali argumenti `init_color()` pa so številke med 0 in 1000, ki povedo, kolikšen del rdeče, zelene in modre barve naj bo v naši novi barvi.

```
// Svetla rdeča barva z nekaj malega zelene in modre
init_color(9, 900, 50, 50);

// Svetla rdeča s črnim ozadjem
init_pair(13, 9, COLOR_BLACK);
```

Seznam ukazov

```
// Inicializacija knjižnice
initscr();
cbreak();
noecho();
keypad(stdscr, true);

// Konec programa
endwin();

// Branje in pisanje
move(y,x);
printw("...");
mvprintw(y,x,"...");
refresh();
getch();
getstr();

// Barve in atributi
attron(atribut);
attroff(atribut);
start_color();
init_pair(stevilka, besedilo, ozadje);
attron(COLOR_PAIR(stevilka));
init_color(stevilka, rdece, zeleno, modro);
```

Seznam barv

Nekatere barve so že na voljo v knjižnici:

```
COLOR_BLACK, COLOR_RED, COLOR_GREEN
COLOR_YELLOW, COLOR_BLUE, COLOR_MAGENTA
COLOR_CYAN, COLOR_WHITE
```

Ostale barve moramo definirati sami s funkcijo `init_color`.

Tipke

```
// Puščice
KEY_UP, KEY_DOWN, KEY_LEFT, KEY_RIGHT
// Brisanje
KEY_DC, KEY_BACKSPACE
// Številke
'0', '1', ..., '9'
// Črke
'a', 'b', ..., 'z', 'A', 'B', ..., 'Z'
// Enter (nova vrstica)
'\n'
// Druge tipke
KEY_HOME, KEY_END, KEY_NPAGE, KEY_PPAGE
```

Uporabne povezave

https://tldp.org/HOWTO/NCURSES-Programming-HOWTO/
https://github.com/mcdaniel/curses_tutorial