```
/*
Minule jsme si ukázali, jak blikat ledkou pomocí bitového komplementu (~).
Nyní bychom se mohli pokusit vytvořit světelného hada. K tomuto účelu se hodí
bitový posun. Jak bylo minule uvedeno, bitový posun se zapisuje >> nebo <<.
Výsledný program by tedy mohl vypadat:
*/

#define F_CPU 1000000UL  // 1 MHz (základní frekvence)

#include <avr/io.h>   //Knihovna vstupů a výstupů (PORT, DDR)
#include <util/delay.h> //Knihovna čekacích funkcí

#define RYCHLOST 10 //Počet kroků za sekundu
#define DELKA    3    //Délka hada (počet ledek))


int main (void)
{
DDRB = 0b11111111;    //piny 0 - 5 budou výstupní
unsigned char n;    //nadefinujeme si proměnnou "n", která bude sloužit
            //k počítání opakování cyklu
PORTB = 0;        //zhasneme všechny ledky. (1=svítí a 0=zhasnuto)

for(;;)
  { //hlavní smyčka

  for(n=0;n<(8+DELKA);n++)  //Smyčka s určitým počtem opakování
                //(8 ledek + délka hada, aby nevznikal
                //nový had, dokud ten starý nezmizí -
                //můžete vyzkoušet, co se stane, když
                //konstantu DELKA smažete.)
    {
    if (n<DELKA)  //pokud je počet opakování menší než délka hada
      {     //musíme přidat další článek hada (rozsvítit led)
            //asi takto: 00000001, 00000011, 00000111, atd.

      PORTB <<=1; //posuneme hada doleva (bity se posunou doleva a
            //vpravo se objeví 0.
            //např: 00001111 po odrotování do leva: 00011110

      PORTB ++; //přičteme 1 (rozsvítíme další článek):
            //00011110 + 1 = 00011111 (hadovi přibyl článek)
      }
    else      //pokud není počet opakování menší než délka hada
      {     //-to znamená, že hada už jsme nakreslili..

      PORTB <<=1; //...tak pouze posuneme doleva
            //(vpravo se automaticky doplní nula)
            // např: 00111110 (had je zakončen, a příště už bude
            //jen "lézt": 01111100, 11111000, 11110000, atd.
      }     //jinak by nám had pořád přirůstal, až by svítily
            //všechny ledky

    _delay_ms (1000/RYCHLOST);  //Čekání (1000 ms = 1 sekunda) - rychlost
                //je tedy počet kroků za sekundu

    } //konec těla cyklu - a znovu na začátek :-)
      //až se cyklus provede 8x (počet ledek) + počet ledek na hada
      //- to znamená, že had už vyleze "mimo ledky"
      //  (10000000 -> 00000000) - všechny ledky jsou zhasnuté
      //- cyklus se ukončí (provedl se "(8+DELKA)" krát)


  } // konec nekonečné smyčky for(;;) program skočí zpátky na začátek
```

```
    // a vše se opakuje.

} //konec funkce main (sem se sice program nikdy nedostane, ale funkce musí
  //být ukončená, jinak by hlásil překladač chybu

/*
V tomto programu jsme použili proměnnou (n), podmíněný blok (if) a smyčku s
určitým počtem opakování (for(n=0;n<8;n++)).
*/
/*
PROMĚNNÁ:
Proměnná je místo v paměti mikrokontroléru, kam se ukládají čísla. Protože
mikrokontrolér potřebuje vědět, jak velký kus paměti si má pro danou proměnnou
vyhradit, píše se při definici proměnné její typ (modrý text před proměnnou).
Přehled typů proměnných:

  TYP:       VELIKOST:  ČÍSLO, JAKÉ JE MOŽNO UCHOVAT:
char         8 bit           -128   až    127
unsigned char    8 bit              0   až    255
int             16 bit         -32768   až    32767
unsigned int    16 bit              0   až    65535
long            32 bit     -2,14*10^9   až    2,14*10^9
unsigned long   32 bit              0   až    4,29*10^9
*/

/*
PODMÍNĚNÝ BLOK:

if (podmínka)
  blok1
else
  blok2

Podmíněný blok slouží k větvení programu. Používá se tehdy, když potřebujeme
provést nějaké příkazy jen tehdy, když platí nějaká podmínka.
Podmínka je výraz, který když je pravdivý, vykoná se blok1 a když pravdivý
není, vykoná se blok2.
Pokud obsahuje blok1 nebo blok2 více příkazů, uzavírají se do složených
závorek "{}".
*/
/*
SMYČKA:

for(inicializátor;podmínka;inkrement)
{
tělo cyklu
}

Inicializátor je vlastně začátek (odkud má cyklus počítat)
Podmínka je výraz, který když platí, tak cyklus provede další opakování.
Pokud výraz přestane platit, cyklus se ukončí a program pokračuje za cyklem.
Inkrement je výraz, který říká, kolik se má přičíst, nebo odečíst s každým
opakováním cyklu. My jsme používali: n++, což je totéž jako: n=n+1. S každým
opakováním cyklu se tedy k proměnné "n" přičte 1.
Tělo cyklu je sada příkazů uzavřená do složených závorek "{}" která se při
každém opakování vykoná.

Dejme tomu, že chceme nějaký blok instrukcí provést 3x.
Napíšeme:
for (n=0;n<3;n++)
{
blok příkazů
}
```

```
Náš cyklus tedy začne:

n=0. Otestuje podmínku n<3 = pravda - pokračuje se. K proměnné "n" se
přičte 1 (n++).
A znovu:
n=1 (0+1=1). Podmínka: n<3 = pravda - pokračuje se. Opět se přičte 1 (n++).
A zas:
n=2 (1+1=2). Podmínka: n<3 = pravda - pokračuje se. Zase se přičte 1 (n++).
A nakonec:
n=3 (2+1=2). Podmínka: n<3 = nepravda - konec cyklu.

Je tedy vidět, že se cyklus provedl 3x.
*/
/*
Nyní se podíváme, co se stalo s registrem PORTB v našem programu.
Nejprve jsme zhasli všechny ledky tak, že jsme do PORTB zapsali samé nuly
(0 = 0b00000000).
Poté jsme vstoupili do hlavní smyčky, která neustále spouští cyklus:
"for(n=0;n<(8+DELKA);n++)". Jakmile se tento cyklus provede tolikrát, kolikrát
je potřeba, program "propadne" zpět do hlavní smyčky "for(;;)" která jej vrátí
zpět na začátek cyklu "for(n=0;n<(8+DELKA);n++)"
Hlavní smyčka má tedy jediný úkol, a to resetovat náš cyklus
"for(n=0;n<(8+DELKA);n++)".
*/

/*
Na začátku programu jsou definovány konstanty "RYCHLOST" a "DELKA", které
určují rychlost pohybu hada a jeho délku. Rychlost je dána počtem kroků za
sekundu (1 - 1000) a délka je dána počtem svítících ledek. (0 - kolik chcete).
*/


//Pro radioklub OK1KVK naspal Vašek Král
```