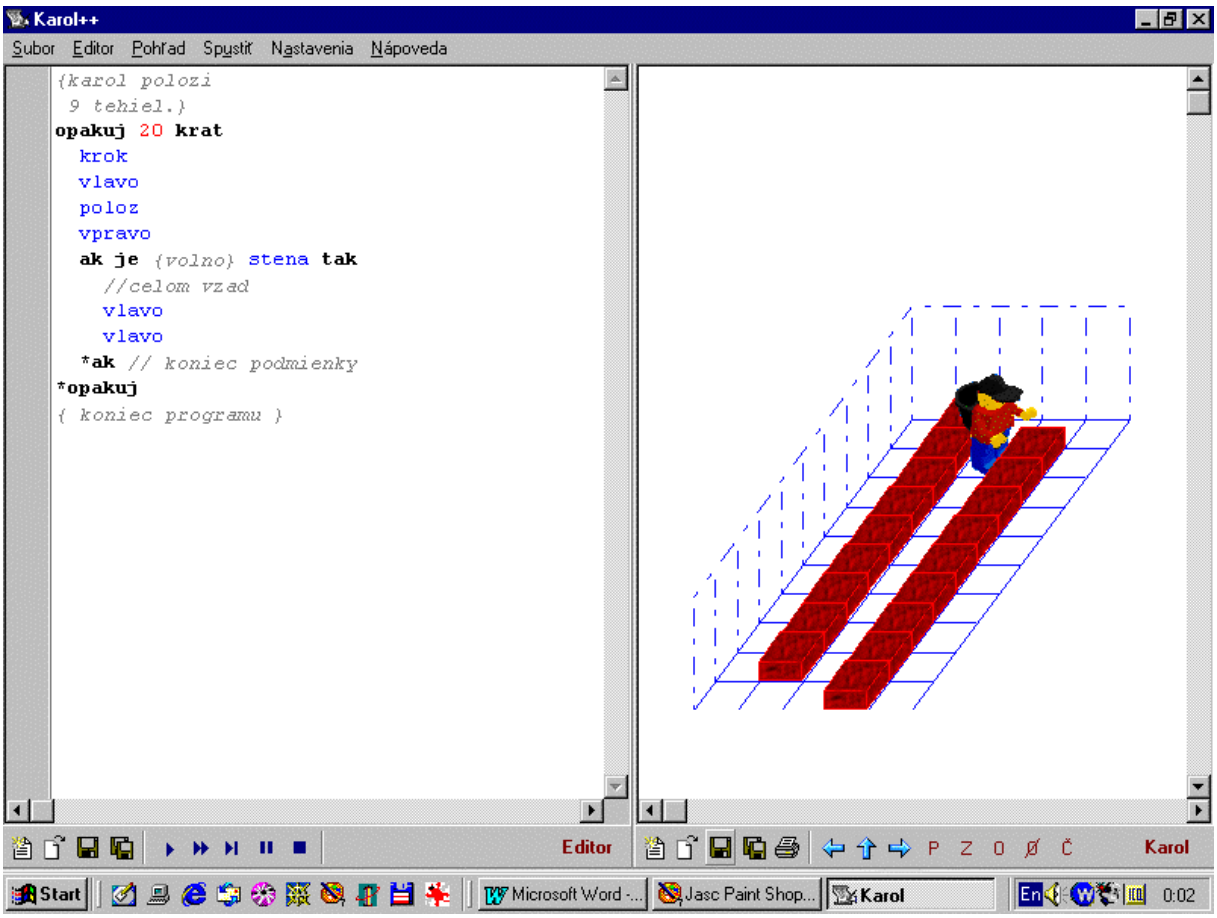
Programy

V tejto časti nájdete ukážkové programy.

Program 1

Prvý program ukazuje použitie cyklu „kým“ a podmienky „volno“. Karol prejde celú miestnosť a zastane pri stene.




```
kym je volno rob
  krok
*kym
```

Skúste si: ostatné cykly, príkazy a podmienky. Skúste napísať program, ktorý prejde po okrajoch miestnosti.

Program 2

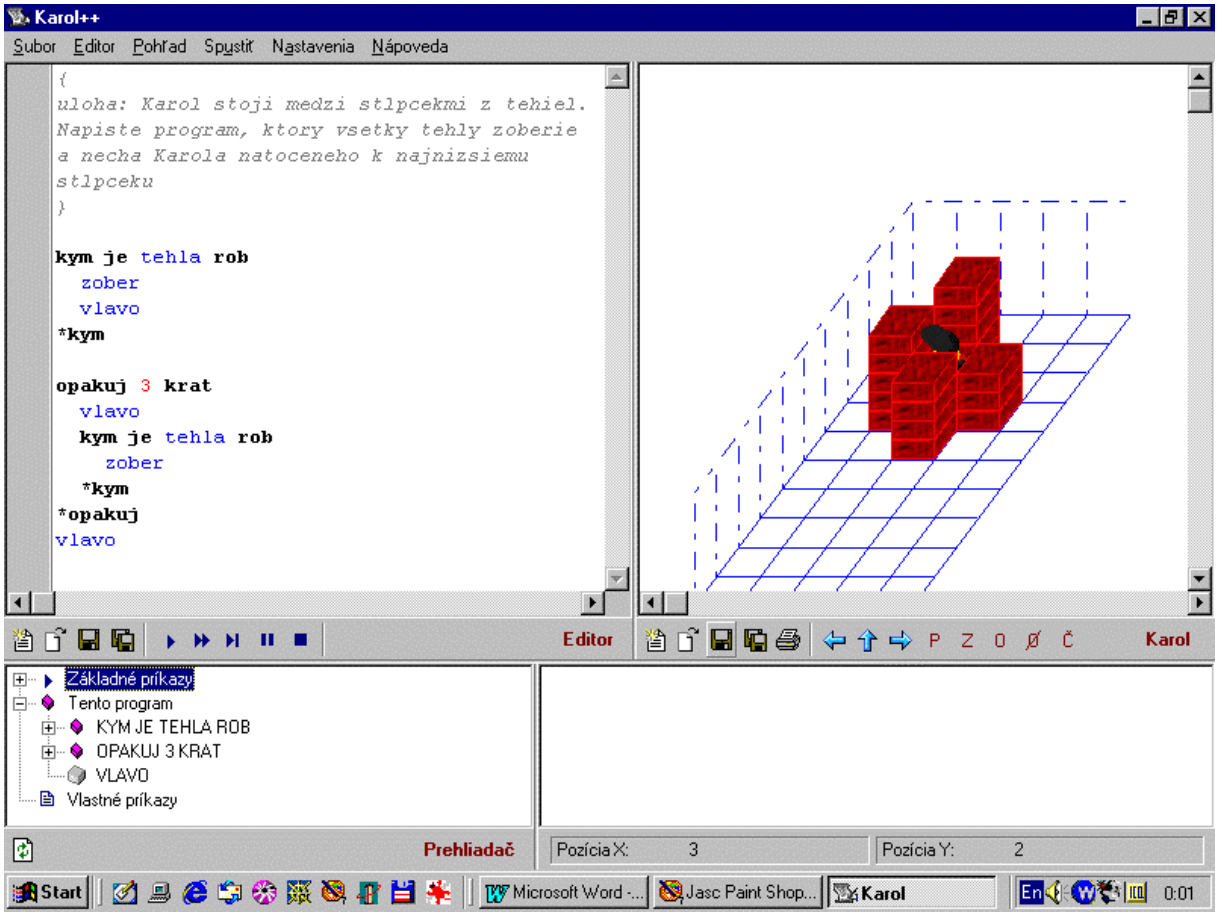
Tento program naučí Karla prejsť po miestnosti a ukladať vedľa seba tehly. Ak Karol príde stene, otočí sa a pokračuje opačným smerom. Príklad ukazuje použitie komentárov (tj. textov medzi zátvorkami {} alebo medzi // a koncom riadka.)

```
{karol polozi
 9 tehiel.}
opakuj 20 krat
  krok
  vlavo
  poloz
  vpravo
  ak je {volno} stena tak
    //celom vzad
    vlavo
    vlavo
  *ak // koniec podmienky
*opakuj
{ koniec
programu }
```

Skúste si: vetvenia Ak. Skúste napísať program, ktorý prejde po okrajoch miestnosti. Program napíšte s použitím jedného cyklu.

Porovnanie počtu tehiel

Úloha: Karol stojí medzi stľpčekmi z tehiel. Napíšte program, ktorý všetky tehly zoberie a nechá Karola natočeného k najnižšiemu stĺpčeku.
Program funguje tak, že Karol postupne odoberá z každého stĺpca, kým nenájde stĺpec, na ktorom už nič nie je. Potom zoberie aj ostatné 3 stĺpce.




```
{
uloha: Karol stoji medzi stlpcekmi z tehiel.
Napiste program, ktory vsetky tehly zoberie
A necha Karola natoceneho k najnizsiemu Stlpceku
}
kym je tehla rob
  zober
  vlavo
*kym

opakuj 3 krat
  vlavo
  kym je tehla rob
    zober
  *kym
*opakuj
vlavo
```

Príkaz celomvzad

Teraz napíšeme príkaz, ktorý otočí Karola o 180 stupňov:

```
prikaz celomvzad
  vlavo
  vlavo
*prikaz
```

Skúste si: Napísať ďalšie príkazy, ktoré budete často používať. Napríklad poloz_vlavo (položí tehlu naľavo od seba)…

Príkaz krokvzad

Teraz napíšeme ďalší príkaz, ktorým naučíme Karola spraviť krok vzad. Príkaz celomvzad je viditeľný len z príkazy krokvzad.

```
prikaz krokvzad

  prikaz celomvzad
    vlavo
    vlavo
  *prikaz

  celomvzad
  krok
  celomvzad
*prikaz
```

Skúste si: ďalšie vnorené príkazy.

Podmienka dvetehly

V Karolovi sa dajú vytvárať aj vlastné podmienky. Táto zistí, či sú na sebe práve dve tehly.

```
Podmienka dvetehly
  Rychlo
  Nepravda
  Ak je tehla tak
     Zober
     ak je tehla tak
        zober
        pravda
        ak je tehla tak nepravda *ak
        poloz
     *ak
     poloz
  *ak
  pomaly
*podmienka

ak je dvetehly tak
  vlavo
inak
  vpravo
*ak
```

Tento program by sa dal riešiť aj omnoho jednoduchšie (pomocou podmienky tehla s parametrom), ale to si ukážeme až neskôr.

Skúste si: napísať podmienku dvetehlyvzadu, ktorá vráti pravdu, ak sú za Karolom dve tehly. Použite vnorený-lokálny príkaz celomvzad.

Rýchlo postav bazén

V nasledujúcom programe nájdete použitie príkazov rychlo a pomaly.

```
// prikaz postav bazen
prikaz postav_bazen;
   {********** postavim bazen **********}
   rychlo
   opakuj 12 krat
     kym nie je stena rob
       poloz
       krok
     *kym
     vlavo
     rychlo
   *opakuj
   pomaly
*prikaz

// prikaz zburaj bazen
prikaz zburaj_bazen;
   {zburam bazen}
   rychlo
   opakuj 12 krat
     kym nie je stena rob
       zober;
       krok;
     *kym
     vpravo;
     rychlo;
   *opakuj
   pomaly;
*prikaz

// prikaz preplavaj bazen
prikaz preplavaj_bazen;

    // lokalne prikazy
    prikaz celomvzad;
       vlavo;
       vlavo;
    *prikaz

  // telo prikazu
   opakuj 3 krat poloz *opakuj
   krok
   kym nie je tehla rob
     opakuj 3 krat poloz *opakuj
     krok
     celomvzad
```

```
     opakuj 3 krat zober *opakuj
     celomvzad
   *kym
   krok
   vlavo vlavo
   opakuj 3 krat zober *opakuj
   vlavo vlavo
*prikaz

// definicia hlavneho prikazu
prikaz hlavny_prikaz;

   postav_bazen;

   {prejdem do stredu}
    vlavo opakuj 2 krat krok *opakuj vpravo

   {preplavaj}
   preplavaj_bazen;

   {pridem do rohu}
   vpravo
   opakuj 2 krat krok *opakuj
   vpravo

   zburaj_bazen;

   {vratim sa na povodnu poziciu}
   kym je volno rob krok; *kym
   {natocenie v povodnom smere}
   vlavo; vlavo;
*prikaz

{ ****************************************** }
{ ************** ZACIATOK PROGRAMU ********** }
{ ****************************************** }
opakuj 4 krat

   hlavny_prikaz;

*opakuj
{ ********** KONIEC PROGRAMU ********** }
```

Rekurzia – prelož stĺpik

Tento program ukazuje použitie rekurzie. Úloha by sa dala urobiť aj bez nej (dokonca aj jednoduchšie). Sú však prípady, keď sa rekurzii nevyhnete.

```
{
uloha: Karol stoji pred stlpcekom z tehiel. Napiste program,
ktory nauci Karla tento stlpcek prelozit za seba.
}

prikaz stlpcek
  ak je tehla tak
     zober
     stlpcek
  inak
     vlavo
     vlavo
  *ak
  poloz
*prikaz

stlpcek
zober
```

Prelož stĺpik bez rekurzie

Prepíšeme predchádzajúci program bez rekurzie. Môžete porovnať rozdiel v tom, ako Karol stĺpik prekladá v porovnaní z príkladom vyššie.

```
kym je tehla rob
  zober
  vlavo
  vlavo
  poloz
  vlavo
  vlavo
*kym
```

Šachovnica

Nasledujúci program rozloží po miestnosti (rozmerov 6*10) značky do šachovnice. Políčko označkujete pomocou príkazu označ a odznačkujete pomocou odznač.

```
prikaz vyznackuj_riadok
  kym je volno rob
    oznac
    krok
    krok
  *kym
*prikaz

opakuj 3 krat
  vyznackuj_riadok
  vlavo; krok; vlavo;
  vyznackuj_riadok
  vpravo; krok; vpravo;
*opakuj
```

Skúste si: napísať program, ktorý naučí Karola robiť šachovnicu v miestnosti ľubovoľných rozmerov.

Parametre príkazov

Niektoré príkazy môžu mať ďalšie parametre. Napríklad podmienka „tehla(5)“ vracia pravdu, ak je pred Karolom práve päť tehiel. Príkaz krok(4) spraví 4 kroky.

```
kym je tehla(3) rob
  krok(2);
*kym
```

Skúste si: napísať program, vďaka ktorému Karol prejde po úplne celej miestnosti a označí všetky stĺpce, kde sú na sebe práve dve tehly.

Sčítavame čísla

Toto je jeden z ukážkových programov. Ukazuje použitie viacerých príkazov jazyka Karol. Program postupne prekladá z jedného stĺpca na druhý tehly (počet tehiel zobrazuje cifru čísla). Ak sa v stĺpci, kam ukladá tehly, objaví 10 tehiel na sebe, Karol ich zoberie a označí políčko pod sebou, čím si zapamätá, že najbližšie tteba dať o jednu tehlu navyše.

```
prikaz celomvzad;
  rychlo
  vlavo
  vlavo
  pomaly
*prikaz

prikaz skontroluj_pretecenie
     ak je tehla(10) tak
       opakuj 10 krat zober *opakuj;
       oznac
     *ak
*prikaz

{prelozi jeden stlpcek na druhy.
 ak sa objavilo "pretecenie", nastavi "flag" - znacku}
prikaz preloz
   kym je tehla rob
     zober
     celomvzad
     poloz
     skontroluj_pretecenie;
     celomvzad
   *kym
*prikaz

{pravdiva, ak ma karol po pravej ruke stenu}
podmienka vpravo_stena
  vpravo
  ak je stena tak pravda inak nepravda *ak
  vlavo
*podmienka

prikaz scitaj_cisla
  //opakuj 9 krat
  kym nie je vpravo_stena rob
    preloz
    vpravo

    ak je znacka tak
      odznac
      krok
      vpravo
```

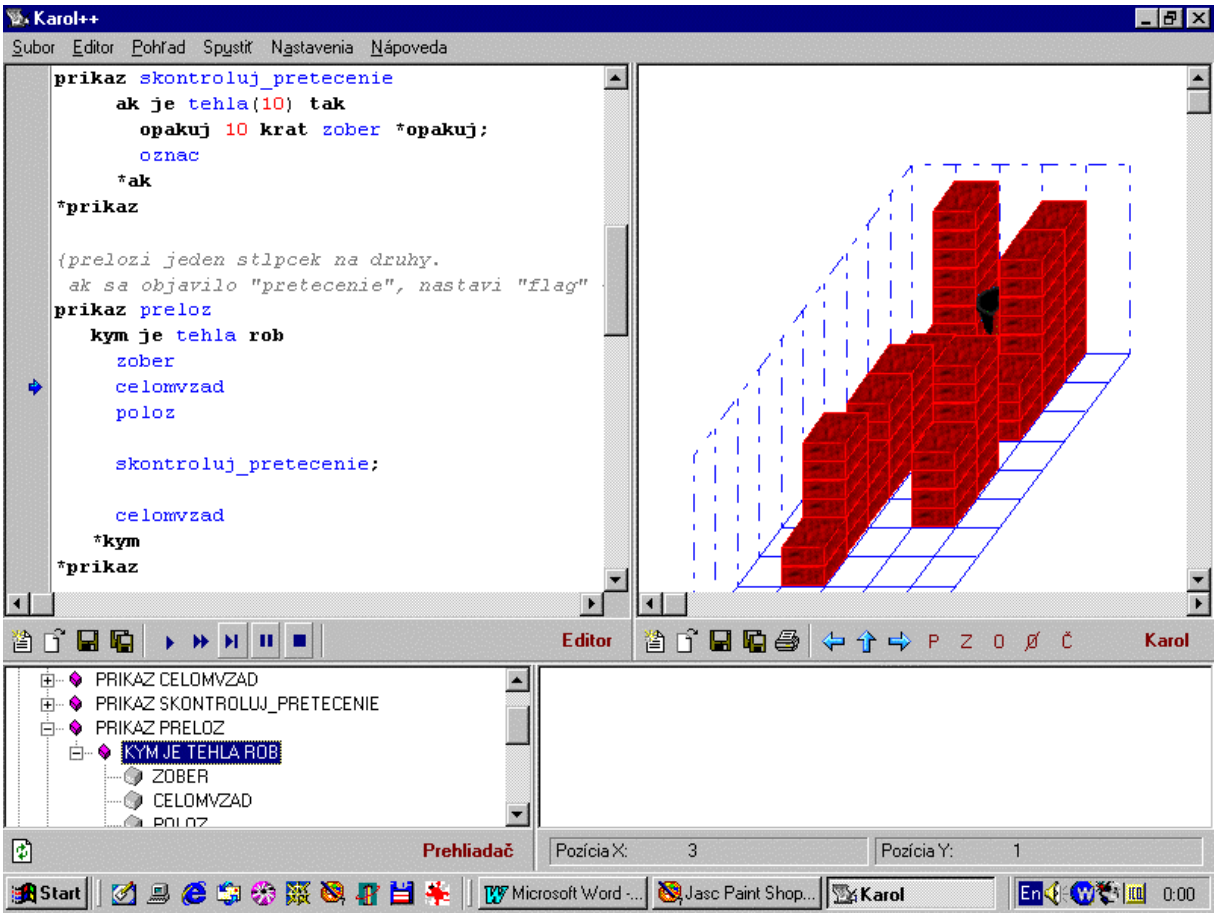
```
      poloz

      skontroluj_pretecenie

      celomvzad
    inak
      krok
      vlavo
    *ak
  *kym
*prikaz

// ZACIATOK PROGRAMU
{otocim karla na dobru stranu}
kym nie je vychod rob vlavo *kym
{scitam cisla}
scitaj_cisla
```

Skúste si: prepísať tento program tak, aby sčítaval čísla v 8-ovej sústave. (stačí urobiť pár úprav)



Prechádzanie bludiska

Toto je posledný ukážkový program. Ukazuje použitie väčšiny vlastností jazyka Karol++. Väčšina z nich bola použitá aj v predchádzajňucich príkladoch, nové sú príkazy PIP a CAKAJ, SKONCI.
Na prechádzanie bludiskom sa používa rekurzia – aby Karol vedel, kam sa má vrátiť.
(v bludisku sú steny reprezentované dvomi tehlami, cieľ sú 4 tehly na sebe).

```
{KAROL SA OTOCI O 180 STUPNOV}
prikaz celomvzad
  vlavo
  vlavo
*prikaz

{KAROL UROBI KROK DOZADU}
prikaz krokvzad
  celomvzad
  krok
  celomvzad
*prikaz

{
******************************************************
**       podmienka vrati pravdu, ak je volno,       **
**     nie su pred karolom znacky ani dve tehly     **
******************************************************
}
podmienka mozno_ist

    {LOKALNA PODMIENKA, VRATI PRAVDU AK JE KROK
    PRED KAROLOM ZNACKA}
    podmienka vpredu_znacka
      krok
      ak je znacka tak
          pravda
      inak
          nepravda
      *ak
      krokvzad
    *podmienka

// telo podmienky mozno ist
nepravda
ak je volno tak
  ak nie je vpredu_znacka tak
    ak je tehla(2) tak
      nepravda
    inak
      pravda
```

```
      ak je tehla(4) tak nepravda *ak
    *ak
  *ak
*ak
*podmienka

{*****************************************************
**         VRATI PRAVDA, AK OKOLO KARLA JE CIEL       **
**                (4 TEHLY NA SEBE)                   **
*****************************************************}
podmienka ciel
  nepravda
  opakuj 4 krat
    ak je tehla(4) tak
      pravda
    *ak
    vlavo
  *opakuj
*podmienka

{*****************************************************
** PRIKAZ, KTORY SA VOLA REKURZIVNE A PRECHADZA       **
** POSTUPNE VSETKY CHODBY                            **
*****************************************************}

prikaz prejdi_vetvu

  // prehlada vsetky mozne cesticky...
  oznac
  opakuj 4 krat
    vlavo
    ak je mozno_ist tak
        krok
        rychlo
        prejdi_vetvu
        { zistim, ci som v cieli,
        inak sa vratim  }
        ak je ciel tak
          { teraz karol pocka 2,5 sekundy,pipne, natoci sa k
            cielu, a potom ukonci program }
          cakaj(2500);
          pip;
          kym nie je tehla(4) rob vlavo *kym
          krok
          pomaly
          skonci // preddefinovany prikaz, ukonci beh programu
        inak
          // vratim sa
          krokvzad
          rychlo
        *ak
```

```
    *ak
  *opakuj

*prikaz
{*************************************************************
***************** TELO PROGRAMU ************************
*************************************************************}
rychlo
prejdi_vetvu
{sem sa karol dostane, len ak sa k cielu neda dostat}
opakuj 2 krat
  pip
  cakaj(2000)
*opakuj
```