Robot Karol++

programovací jazyk pre deti

popis jazyka
popis prostredia
okomentované a slovne popísané príklady

Ondrej Krško
www.gjh.sk/~krsko/karol

Robot Karol++

Robot Karol++ je:
programovací jazyk (pozrite si popis jazyka) - je to programovací jazyk určený na výučbu programovania pre deti.
programovacie prostredie, ktoré zvýrazňuje syntax, doplňuje slová, obsahuje "debugger" umožňujúci krokovať program, zobrazuje program v Prehliadači kódu...

Testovanie

Testovanie prebiehalo na viacerých počítačoch. Na žiadnej konfigurácii neboli nájdené žiadne chyby:

- Intel Pentium 75 Mhz, 8 Mb RAM, Windows 95
- Intel Pentium 120 MHz, 16 Mb RAM, Windows 95
- Intel Pentium 133 MHz, 48 Mb RAM, Windows 98SE
- Intel Pentium II, 64 Mb RAM, Windows NT Service Pack 5
- Intel Pentium Celeron 800Mhz, 64 Mb RAM, Windows 2000 Profesional
- Intel Pentium III, 512 Mb RAM, Windows 2000 Server Service Pack 1

Jazyk

Štandartný jazyk Robota Karla:
- Preddefinované príkazy
- Vlastné príkazy
- Preddefinované podmienky
- Vlastné podmienky
- Cyklus kým
- Cyklus opakuj
- Vetvenie Ak
- Rychlo / pomaly

Rozšírenia jazyka Karol++
- Základné rozšírenia

Preddefinované príkazy

Karol pozná nasledujúce preddefinované príkazy:

krok - robot urobí krok v smere, v ktorom je natočený
vlavo - otočí sa doľava
vpravo - otočí sa doprava
poloz - položí pred seba tehlu
zober - zoberie tehlu spred seba
oznac - robot dá pod seba značku
odznac - Karol zoberie značku spod seba
cakaj - Karol pocka jednu sekundu, mozno zadat aj "cakaj(*poc_milisekund*)".
pip - Karol "pipne".

Vlastné príkazy

Možete vytvárať aj vlastné príkazy. Napísať ich možno priamo v programe, tak ako to je aj v iných jazykoch.

```
{zaciatok prikazu}
prikaz poloz3tehly
  opakuj 3 krat
    poloz
  *opakuj
*prikaz
{koniec prikazu}

{zaciatok programu}
kym je volno rob
  poloz3tehly
  krok
*kym
{koniec programu}
```

Preddefinované podmienky

Karol pozná tieto preddefinované podmienky:

stena - vráti PRAVDU, ak Karol stojí pri stene a je k nej otočený čelom.
volno - vráti PRAVDU, ak môže Karol spraviť krok bez toho, aby narazil
tehla - vráti PRAVDU, ak je pred Karlom tehla (je k nej natočený čelom)
znacka - vráti PRAVDU, ak je pod Karlom znacka
juh, **sever**, **zapad**, **vychod** – vrátia PRAVDU podľa natočenia Karola v miestnosti.

Vlastné podmienky

Okrem preddefinovaných podmienok môžete používať aj vlastné. V definícii podmienky môžete používať aj slová: PRAVDA, NEPRAVDA určujúce návratovú hodnotu podmienky.

```
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
```

Cyklus kým

Cyklus kym má nasledujúcu štruktúru:

```
kym (nie) je <podmienka> rob
  <prikazy>
  ...
  <prikazy>
*kym
```

<podmienka> - podmienka môže byť vlastná alebo preddefinovaná
<prikazy> - ľubovoľné príkazy a ďalšie cykly.

Cyklus opakuj

Cyklus opakuj má nasledujúcu štruktúru:

```
opakuj <X> krat
  <prikazy>
  ...
  <prikazy>
*opakuj
```

<X> - celé číslo, počet opakovaní
<prikazy> - ľubovoľné príkazy a ďalšie cykly...

Vetvenie ak

Vetvenie ak má nasledujúcu štruktúru:

```
ak (nie) je <podmienka> tak
  <prikazy>
  ...
  <prikazy>
*ak
```

alebo

```
ak (nie) je <podmienka> tak
  <prikazy>
  ...
  <prikazy>
inak
  <prikazy>
  ...
  <prikazy>
*ak
```

<podmienka> - vlastná alebo preddefinovaná podmienka
<prikazy> - ľubovoľné príkazy a ďalšie cykly, vetvenia...

Naraz / pomaly

Ak chceš nechať Karola pracovať naraz (aby príkazy nevykonával postupne), zadaj mu slovo "rychlo". Slovom pomaly ho prepneš do pôvodného módu.

```
{
V tomto priklade su pouzivane
vlastne prikazy a vnorene prikazy
  (prikaz preplavaj_bazen ma este
  prikaz celomvzad)
}

// 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 ********** }
```

Rozšírenia

oddelovače - okrem medzery, prázdneho riadku možno použiť aj bodkočiarku podobne tak, ako v jazyku Pascal, C, C++

```
opakuj 10 krat
  vlavo; poloz; vpravo;
  vpravo; poloz; vlavo;
  krok;
*opakuj
```

komentáre - časti v programe, ktoré sa nevykonávajú. Slúžia na sprehľadnenie programu. Možno používať jednoriadkové komentáre a viacriadkový komentár

```
opakuj 10 krat
  vlavo; poloz; vpravo; // karol polozi tehlu nalavo od seba
  vpravo; poloz; vlavo;
  {
    teraz ma karol okolo seba
    dve tehly - jednu pri lavej
    ruke a druhu pri pravej
  }
  krok;
*opakuj
```

vnorené príkazy a podmienky – každý príkaz/podmienka môže obsahovať v svojom tele ďalšie príkazy/podmienky. Tie opäť môžu obsahovať ďalšie.

```
opakuj 10 krat
  vlavo; poloz; vpravo; // karol polozi tehlu nalavo od seba
  vpravo; poloz; vlavo;
  {
    teraz ma karol okolo seba
    dve tehly - jednu pri lavej
    ruke a druhu pri pravej
  }
  krok;
*opakuj
```

parametre príkazov – existujú príkazy, ktorým možno zadávať rozširujúce vlastnosti.

```
Ak je tehla(10) tak {urcuje, ci je na sebe prave 10 tehiel}
  Opakuj 10 krat
    Zober
  *opakuj
*ak
```
