KSI

Korespondenční Seminář z Informatiky

KSI, neboli Korespondenční Seminář z Informatiky, je celoroční soutěž pro středoškoláky organizovaná studenty Fakulty Informatiky Masarykovy Univerzity. Cílem semináře je seznámit řešitele se zajímavými oblastmi informatiky a procvičit programátorské a matematické myšlení. Chcete se naučit nové věci, potrénovat svůj mozek, zasoutěžit si, nebo jet na zážitkové soustředění? Začněte řešit KSI!

Každý ročník semináře se skládá z pěti tematických sad. Sada vždy začíná úvodním povídáním, ve kterém vás, řešitele, uvedeme do tématu, vysvětlíme základní myšlenky, pojmy a triky. Následují soutěžní úlohy různého charakteru, na jejichž vyřešení a sepsání řešení máte zhruba jeden měsíc. Zatímco nejjednodušší úlohy by po přečtení úvodního povídání neměly dělat problémy ani úplným začátečníkům, nad tou nejtěžší se může zapotit i ostřílený řešitel programátorských soutěží.

Zadání zveřejňujeme na našich internetových stránkách a řešitelům je zasíláme i poštou. Odevzdávání úloh a další komunikace pak probíhá přes náš webový systém. Po termínu odevzdání vaše řešení opravíme, obodujeme a okomentujeme. Po vyhodnocení poslední sady úloh jsou nejúspěšnější řešitelé pozváni na týdenní soustředění. Úspěšní řešitelé semináře (ti, kteří dosáhnou alespoň 60% celkového počtu bodů) jsou zároveň přijati na Fakultu informatiky MU bez přijímacích zkoušek. Více informací hledejte na

`http://ksi.fi.muni.cz`

Jaké budou úlohy? Jak psát a odevzdat řešení?

Ve většině úloh bude vaším úkolem řešit algoritmický či programátorský problém. Nikdy nebudete potřebovat znát nějaký konkrétní programovací jazyk. Některé z našich úloh mohou být složité nebo pracné, ale s dobrým nápadem se většinou dají vyřešit rychle a elegantně. Není třeba vyřešit všechny úlohy sady – pošlete, co vyřešíte, zbytek si po uzavření sady můžete přečíst ve vzorovém řešení. Také není nutné, abyste měli úlohu kompletně dořešenou. I pokud máte jen nějaké nápady co s ní, napište nám je a my vaši snahu určitě oceníme.

Pokud sepisujete řešení úloh poprvé, určitě se podívejte na článek „Jak psát řešení“ na našem webu. Zde zdůrazníme, že při opravování hodnotíme především správnost postupu (funkčnost) a efektivitu řešení, ale v neposlední řadě také kvalitu popisu a zdůvodnění správnosti. Pozor, u algoritmických úloh letos nově **vyžadujeme pseudokód**, tedy strukturovaný zápis algoritmu, který by měl být snadno pochopitelný a mělo by z něj být jasné, jak by se algoritmus implementoval v nějakém programovacím jazyce. V článku „Pseudokód“ na našem webu najdete podrobnější vysvětlení a několik příkladů zápisu algoritmu v pseudokódu.

Řešení se odevzdávají elektronicky na našich stránkách `http://ksi.fi.muni.cz`. Preferujeme texty ve formátu **PDF** (`.pdf`), dále přijímáme i formáty `.doc`, `.odt`, `.txt`. Nepřijímáme nový formát Wordu `.docx`, převeďte vaše řešení do PDF. V textu řešení prosím vždy **uveďte své jméno a školu**.

Co vás čeká v první sadě letošního KSI? Seznámíte se s robotem Karlem, pomůžete mu najít cestu bludištěm a vyřešit několik dalších netradičních problémů. Je před vámi tedy série programátorských logických úloh. Vaším cílem bude naprogramovat robota Karla, který se pohybuje na zadaném plánku, tak, aby splnil nějaký úkol. K dispozici budete mít grafický simulátor, abyste si vše mohli vyzkoušet.

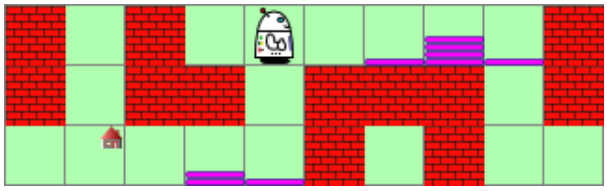
Robot Karel

Karel je jednoduchý, výukový programovací jazyk. Pomocí příkazů a funkcí ovládáme robota Karla, který se pohybuje na plánku v podobě čtvercové sítě. Interpret příkazů, který budeme používat v této sadě, najdete na `http://ksi.fi.muni.cz/karel`.

Plánek Robot se pohybuje na obdélníkovém plánku, který může mít různé velikosti. Na některých polích jsou *zdi*, na ty Karel lézt neumí. Ostatní pole jsou *průchozí*. Robot nemůže vyjít z plánku pryč. Můžeme si tedy pro jednoduchost představit, že plánek je ohraničen zdí.

Na každém průchozím políčku mohou být uloženy *značky* (v prostředí značené jako fialové cihličky). Karel umí značky na políčka pokládat, sbírat je, nebo se podívat, jestli je na políčku nějaká značka. Značky jsou sice jen jednoho typu, ale na jednom políčku jich může být *libovolný počet*.

Jedno políčko v plánku je označeno jako *domov*, zde robot typicky začíná konat program.



Základní příkazy

Karel je robot a jako takový plní příkazy, které dostane od programátora. Moc chytrosti ale bohužel nepobral, takže umí jen následující čtyři instrukce.

KROK
: Robot provede jeden krok vpřed ve směru, v jakém je natočen. Pokud to nelze provést, protože je před Karlem zeď, nebo konec plánku, robot havaruje.

VLEVO-VBOK
: Otočí se o 90° proti směru hodinových ručiček.

POLOŽ
: Robot položí jednu značku na políčko, kde se aktuálně nachází.

ZVEDNI
: Robot zvedne jednu značku z políčka, kde se aktuálně nachází. Pokud tam žádná není, zhavaruje.

Pomocí těchto základních příkazů již umíme Karla prohánět po plánu. Všimněte si, že robot je tak hloupý, že se sám neumí otočit vpravo. Jak ale uvidíme dále, programátor mu ale může nadefinovat nový příkaz **VPRAVO-VBOK**, který se skládá ze třech příkazů **VLEVO-VBOK** a otočí tedy robota doprava.

Podmíněné příkazy

Jen se zmíněnými základními příkazy by Karel nikoho nezaujal. Naštěstí to ale není vše! Aby mohl robot reagovat na různé situace, umí se rozhodovat podle svého okolí. Těmto konstrukcím běžně říkáme *podmínky* a *cykly*.

KDYŽ *podmínka* příkazy 1 **KONEC, JINAK** příkazy 2 **KONEC**
: Karel prozkoumá, jestli je podmínka splněna. Pokud zjistí, že ano, provede příkazy 1, a pokud ne, provede příkazy 2. Uvedení slov KONEC, JINAK následovaných příkazy 2 není povinné a může být vynecháno.

DOKUD *podmínka* příkazy **KONEC**
Karel zjistí, jestli je podmínka splněna a dokud bude platit, bude provádět příkazy.

OPAKUJ *počet* příkazy **KONEC**
Karel bude opakovat příkazy tolikrát, kolik je uvedeno v celočíselném parametru počet.

Slovo „příkazy“ označuje libovolnou posloupnost příkazů, které Karel zná. To mohou být buď základní příkazy Karla, nebo funkce vytvořené programátorem. Karel umí následující 4 podmínky. Každá má dvě varianty: JE a NENÍ. Slovo JE není třeba uvádět (podmínka je defaultně v kladném tvaru).

JE/NENÍ ZEĎ
Podmínka je splněna, pokud před Karlem je/není zeď. „Před Karlem“ znamená vedle políčka na kterém robot stojí ve směru, kterým je natočen.

JE/NENÍ ZNAČKA
Podmínka je splněna, pokud je na políčku, na kterém robot stojí, alespoň jedna značka.

JE/NENÍ DOMOV
Jedno políčko v plánu je označeno jako DOMOV. Tato podmínka je splněna, jen pokud Karel stojí na tomto poli.

JE/NENÍ SEVER
Podmínka je splněna, je-li Karel natočen na sever (tj. nahoru).

Funkce

Příkazy pro Karla můžeme shlukovat do funkcí. Funkce je logická jednotka programu, který plní nějaký úkol. Definováním funkce učíme Karla nové příkazy. Syntaxe je jednoduchá. Na prvním řádku uvedeme jméno nové funkce. Pak následuje posloupnost příkazů, která je ukončena slovem KONEC.

Typická funkce vypadá například takto:

```
DOJDI KE ZDI
   DOKUD NENÍ ZEĎ
      KROK
   KONEC
KONEC
```

Pokud chceme „udělat program“ pro Karla, který plní nějaký úkol, myslíme tím nadefinovat příslušnou funkci.

Rekurze

Rekurze je proces, kdy nějaká funkce volá(tj. používá) sama sebe, případně se více funkcí volá navzájem. Předchozí příklad je možné zapsat pomocí rekurze například následujícím způsobem.

```
DOJDI KE ZDI
    KDYŽ NENÍ ZEĎ
        KROK
        KE ZDI
    KONEC
KONEC
```

```
DOJDI KE ZDI A ZPĚT
    KDYŽ JE ZEĎ
        VLEVO-VBOK
        VLEVO-VBOK
    KONEC, JINAK
        KROK
        KE ZDI
        KROK
    KONEC
KONEC
```

Myšlenka v příkladu nalevo je jednoduchá – pokud chceme dojít ke zdi, zkontrolujeme, jestli u ní už nejsme. Pokud u ní už jsme, končíme. Pokud ne, uděláme krok a pokračujeme stejným způsobem. Nepotřebujeme tedy cyklus, stačí, když funkce zavolá sama sebe.

Příklad vpravo je malinko komplikovanější. Sami si vyzkoušejte, jak se robot chová. Trik je v tom, že funkce volá sama sebe uprostřed bloku příkazů. Poté, co se vnitřní volání funkce dokončí, pokračuje se ve vykonávání vnější funkce a provedou se zbylé příkazy. Tento příkaz tedy zajistí, že Karel dojde ke zdi, tam se otočí a vrátí se zpět přesně na stejné pole, kde začal.

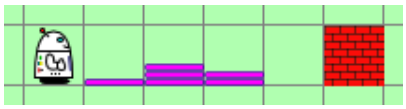
Rekurze je velmi silný nástroj a jak vidíme v předchozím příkladě, může sloužit jako jakási „paměť“ pro robota. Její použití si jistě sami vyzkoušíte v úlohách první sady.

Ukázkový problém

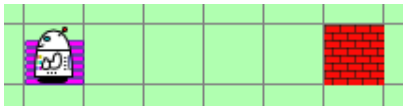
Uvažujme následující jednoduchou úlohu. Úkolem robota je jít k nejbližší zdi v jeho směru, „sesbírat“ všechny značky po cestě a nanosit je na svoje startovní pole. Na konci programu musí robot skončit na svém startovním poli, jeho směr může být libovolný.

Úkol nejlépe ilustruje následující obrázek:

Před:



Po:



Vaše řešení by mohlo vypadat například následovně. Naprogramuji rekurzivní funkci SESBÍREJ, která plní daný úkol, přičemž robot pro provedení funkce stojí v opačném směru než na začátku. Pokud robot stojí před zdí, pouze se otočí a úkol je splněn. Pokud nestojí před zdí, můžeme udělat krok a zavolat rekurzivně funkci SESBÍREJ. Ta nanosí všechny značky na pole, na kterém nyní stojí robot. Zbývá všechny značky přesunout o jedno pole zpět, na původní pozici robota. To lze udělat jednoduchým cyklem. Program je uveden napravo (funkce VZAD je definována jako dvakrát VLEVO-VBOK).

```
SESBÍREJ
  KDYŽ NENÍ ZEĎ
    KROK
    SESBÍREJ
    DOKUD JE ZNAČKA
      ZVEDNI
      KROK
      POLOŽ
      VZAD
      KROK
      VZAD
    KONEC
    KROK
  KONEC, JINAK
    VZAD
  KONEC
KONEC
```

Vývojové prostředí

Své programy si můžete vyzkoušet ve webovém prostředí na adrese

`http://ksi.fi.muni.cz/karel`.

Jeho použití je poměrně intuitivní a snad s ním nebudete mít problém. Příkazy můžete psát malými i velkými písmeny, s diakritikou i bez. V případě jakýchkoli problémů klikněte na odkaz „Nápověda“, kde je vše podrobně popsáno.

Děkujeme panu Oldřichu Jedličkovi, který je autorem webového interpretu robota Karla, jehož upravenou verzi používáme. Původní prostředí najdete na adrese `karel.oldium.net`.

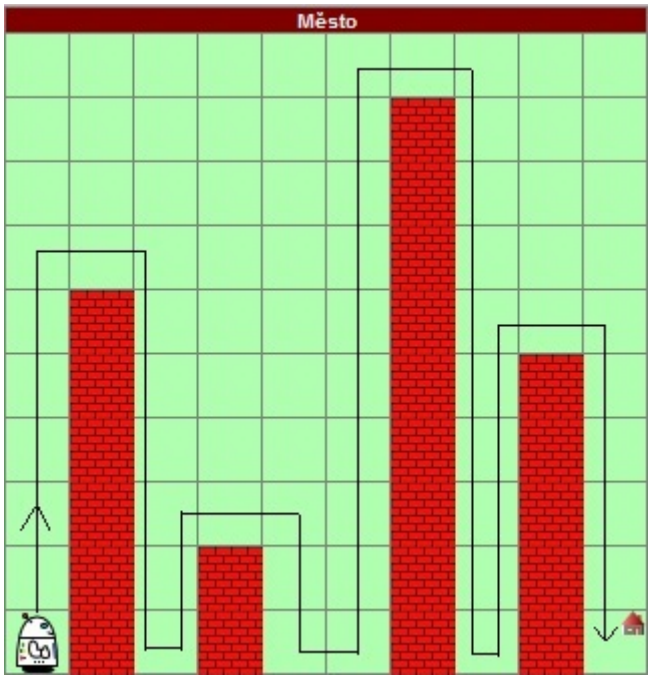
Zadání 1. sady úloh KSI (termín odevzdání: 14. 10. 2012)

K popisu vašeho řešení prosím připojte i textový výpis programu, který najdete dole v simulátoru. Svá řešení nezapomeňte před odevzdáním otestovat. Řešení zasílejte pomocí internetového systému na adrese `http://ksi.fi.muni.cz`.

Příklad 1: Ploty, hory a stopovanie (10 bodů)

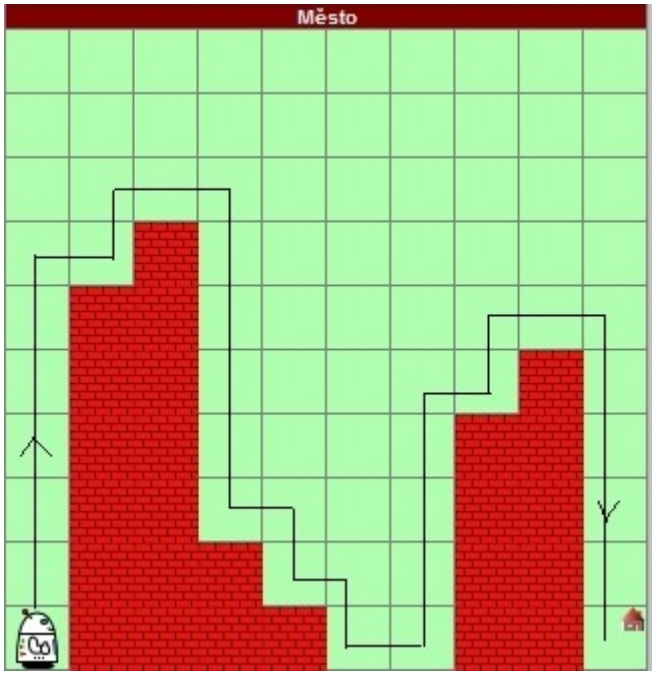
Na zahriatie sme si pre Vás pripravili tri jednoduché úlohy.

a) Ploty V prvej časti je Vašou úlohou naprogramovať Karla tak, aby postupne popreliezal ploty, ktoré mu stoja v ceste domov. Každý plot je široký maximálne 1 stĺpec a medzery medzi jednotlivými plotmi sú široké minimálne jeden stĺpec, pričom ploty nemusia byť rovnako vysoké. Žiaden z plotov sa však nedotýka horného okraja mesta. Karel nie je žiaden skokan, a preto sa pri preliezaní musí vždy pridržiavať plota (viď obrázok).



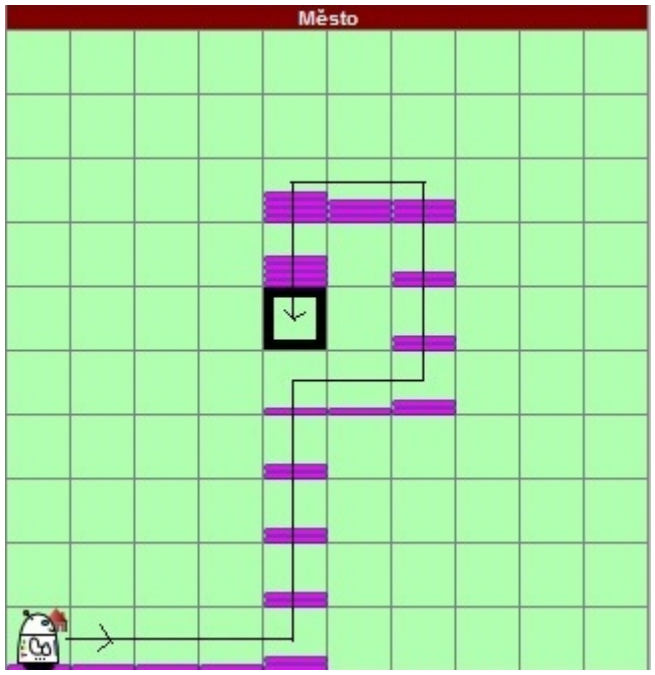
b) Karel horolezcom

V druhej časti je Vašou úlohou opäť dostať Karla domov, tentokrát mu však v ceste stojí obrovské pohorie, cez ktoré musí Karel preliezť. Samotné pohorie sa skladá z rôzne vysokých stĺpcov medzi ktorými môže, ale aj nemusí byť medzera.



c) Stopovanie

V tretej časti pomôžete Karlovi so stopovaním jeho nezvestného kamaráta Jozefa. Ten za sebou počas svojej cesty zanechal značky, ktoré vyznačujú smer ktorým sa z každého políčka pohol. Použil pri tom nasledujúce pravidlá: 1 značka znamená, že Jozef urobil krok na východ, 2 značky znamenajú krok na sever, 3 značky krok na západ a 4 značky krok na juh. Vašou úlohou je zbierať Jozefove značky, nasledovať ich a nájsť miesto, na ktorom sa naposledy nachádzal. To ľahko spoznáte podľa toho, že to bude prvé políčko bez značiek na ktoré narazíte.



Příklad 2: Poetická (10 bodů)

Přestože programovací jazyk Karel vznikl ve Spojených státech, byl pojmenován podle českého spisovatele Karla Čapka, v jehož hře *R.U.R.* se poprvé objevilo slovo „robot“[1]. Jedná se o jednu z mnoha spojitostí mezí literaturou a programováním.

Ostatně zdrojové kódy můžeme označit za formu literárního díla a ne nadarmo se programování považuje za umění. Asi nejvíce se programování podobá poezii.[2] Moderní básně mají se zdrojovými kódy společné i to, že bez komentářů nejdou pochopit. Doufám však, že zadání Vašeho úkolu pochopíte snadno:

Člověk je tvor dosti líný
Pořád by se jenom válel
Ani já v tom nejsem jiný
Úkoly ať dělá Karel!

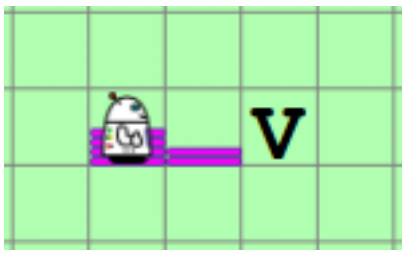
Musel bych tedy uvítat
A body za to přidělit
Kdyby Karel uměl počítat
A já tak měl volnou neděli

Násobení by měl zvládat
Toť Váš úkol druhý
Musím svoje zkoušky skládat
Na matiku chci sluhy

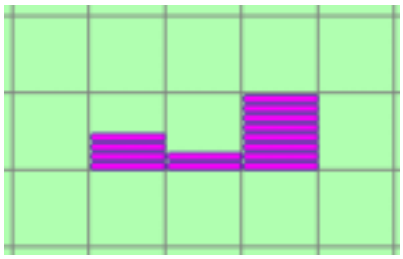
Orgové jsou proradní
Takže abyste si zvykli
Úkol Vám snad znesnadní
Zákaz používat cykly

Přesné zadání

Na dvou sousedních polích je určitý počet značek. Označme tyto počty jako n a m. Karel začíná na jednom z těchto polí a je otočený směrem k druhému poli se značkami. Po provedení Vašeho programu by na políčku označeném na obrázku písmenem V (na které by ze své startovní pozice došel pomocí dvou kroků vpřed) mělo být právě $n \cdot m$ značek. Můžete předpokládat, že tento součin je menší než maximální počet značek, který se do políčka vejde. Políčka s m a n značkami může Váš program měnit - není nutné, aby po skončení programu byla ve stejném stavu jako na začátku. V programu však nesmíte používat žádné cykly (`DOKUD`, `OPAKUJ`).

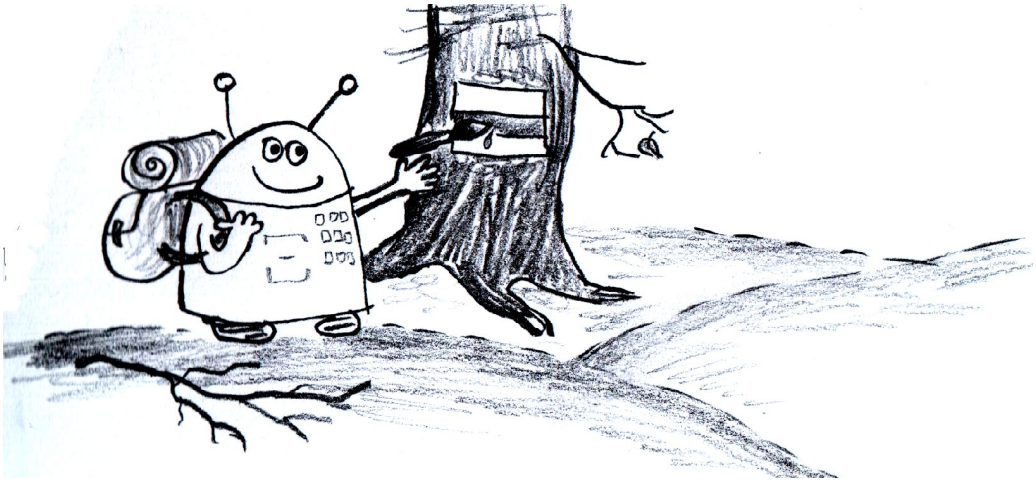


Obr. 1: Výchozí situace.


Obr. 2: Výsledek.

[1] I když toto slovo vymyslel jeho bratr Josef Čapek.

[2] A mimochodem: Za první programátorku je považována dcera *básníka* George Gordona Byrona. I její křestní jméno (Ada) bylo použito jako název programovacího jazyka.

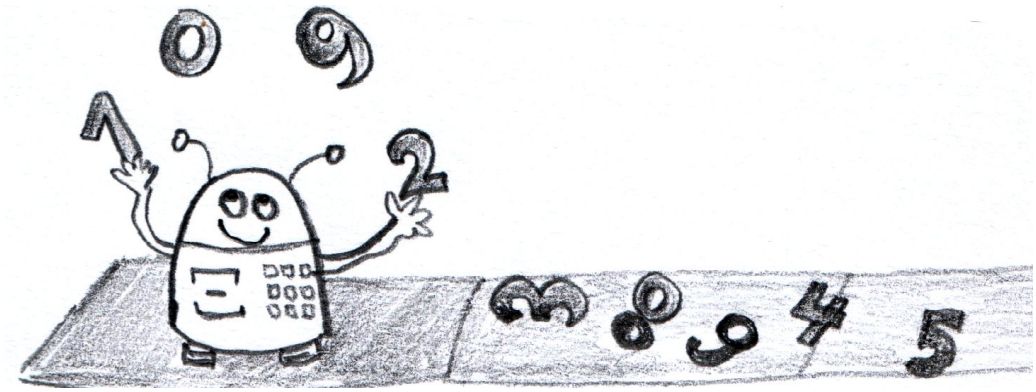


Příklad 3: Návštěva (10 bodů)

Robot Karol sa raz rozhodol, že by ho mohol prísť navštíviť robot Alfonz. Na tom by nebolo nič divné, no má to jeden malý háčik. Karol vždy, keď sa niekam zatúla, tak mu potom nevadí hľadať cestu domov aj veľmi dlho (rozumej náhodne chodí, a raz snáď trafí). Alfonz je však iný, a veľmi ho rozčuluje, keď musí pri ceste zbytočne zachádzať do slepých uličiek. Karol preto chce pre Alfonza vyznačiť cestu, po ktorej sa z miesta, kde práve stojí, dá dostať do domčeka.

Vyznačená cesta nemusí byť najkratšia, no nesmie zachádzať do slepých uličiek (tzn ak robot pôjde podľa tejto cesty, nebude musieť obrátiť sa smerom nazad, lebo všade okolo neho sú steny alebo políčka, na ktorých už bol). Vyznačenie musí byť sledovateľné robotom Alfonzom s jednoduchým programom, napríklad ak je celá cesta vyznačená konkrétnym počtom tehličiek, iným než zvyšok plánika.

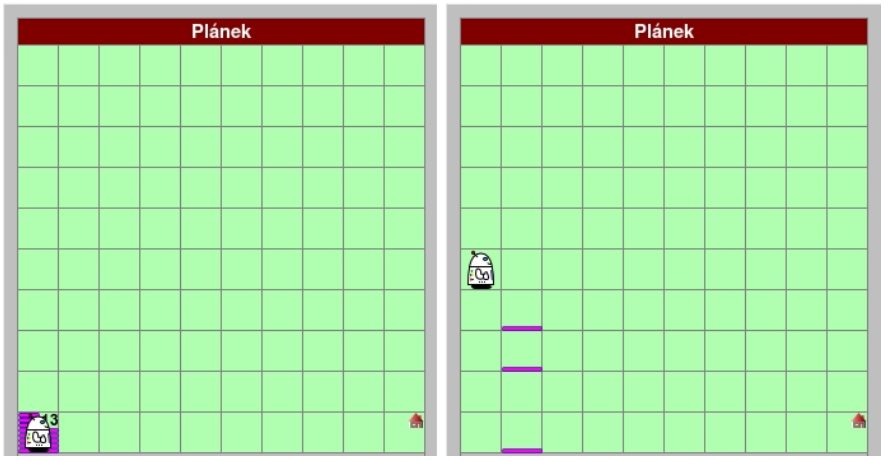
Ak bude program pre Alfonza na sledovanie cesty zložitejší, ako spočítanie tehličiek oproti nejakému danému počtu, tak napíšte do riešenia aj ten.



Příklad 4: Binární (10 bodů)

Na plánku je Karel v levém dolním rohu na políčku, kde je N značek. Cílem je zapsat N v binární soustavě jako posloupnost nul a jedniček do vedlejšího sloupce plánku. Pomocí žádné nebo jedné značky na příslušném políčku "zapíšeme"0 nebo 1. Viz obrázek před a po výpočtu pro $N = 13$, což je v binární soustavě 1101 protože

$$N = 13 = 8 + 4 + 1 = 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0.$$

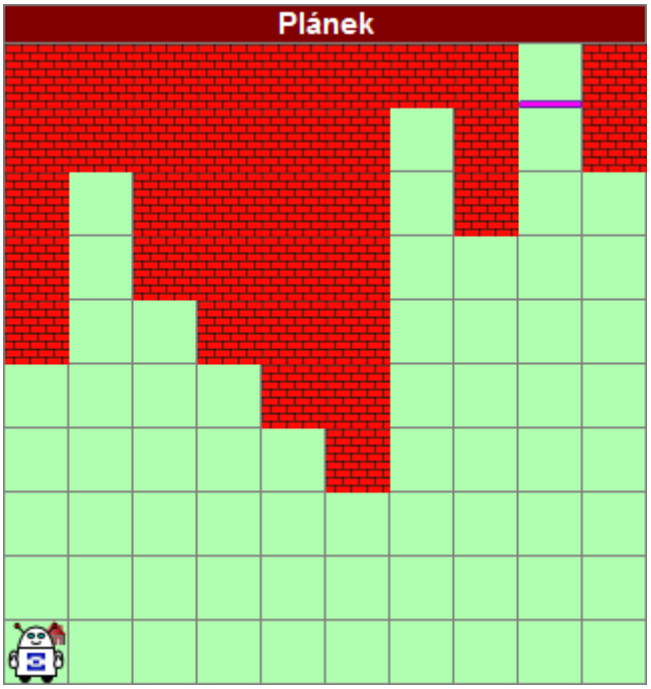


Příklad 5: Karel ve velkoměstě (10 bodů)

Robot Karel se přestěhoval do velkoměsta. V každém sloupci tam stojí jeden mrakodrap, nad ním je už jenom obloha (která shodou okolností v kybernetickém světě vypadá úplně stejně jako zeď). Karel by si rád prohlédl celé město hezky z výšky a rozhlédl se z nejvyššího mrakodrapu. Neví ale, který z nich to je. Pomůžete mu to vypočítat?

V každém sloupci stojí právě jeden mrakodrap výšky minimálně 2 políčka, ve městě je právě jeden mrakodrap, který je nejvyšší. Všechny mrakodrapy mají šířku jeden sloupec a stojí kolmo (tj. nezasahují do sousedních sloupců). Nebe není děravé a nad nebem je proto vždy zase jenom nebe. Karel se přistěhoval do mrakodrapu nula do nultého patra. Město je zprava ohraničeno oblohou v úrovni 0.

BONUS: Dokážete Karla naprogramovat tak, aby pracoval správně i ve městě, kde mají mrakodrapy minimální výšku pouze jedno patro a nejvyšších mrakodrapù může být více než jeden (musí označit všechny)?



Příklad vyřešené úlohy 5. Nejvyšší mrakodrap je ve sloupci 8 a má označené vrchní patro právě jedním tokenem. Všechna ostatní pole jsou bez tokenů.

A to je z první sady KSI vše. Přejeme ti hodně úspěchů při řešení, a když budeš mít jakékoliv otázky, neváhej se na nás obrátit e-mailem na adresu `ksi@fi.muni.cz` nebo v diskuzním fóru na webových stránkách.

Termín odevzdání 1. sady úloh KSI: 14. 10. 2012

`http://ksi.fi.muni.cz`