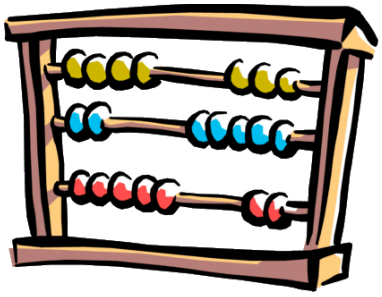
Úloha č. 5
Kuličkový počítač #2



Zamysli se! – seriálová úloha 10 b

Tato úloha je teoretického rázu, tvým úkolem zde není napsat program v klasickém slova smyslu. Namísto toho se po tobě chce vytvořit obrázek (diagram) programu pro kuličkový počítač dle specifikace níže. K obrázku také přilož popis základní myšlenky, na které je program založen.

Tato úloha je seriálová, což znamená, že se s kuličkovým počítačem budeš setkávat po celý letošní ročník FIKSu. V každém z následujících kol využiješ znalosti z kol předcházejících. Právě proto je pro úspešné vyřešení této úlohy zapotřebí znát prostředky pro popis programů, které jsou popsány v úloze Kuličkový počítač #1 z předchozího kola.

Spolu s úspěšně probíhající vzpourou se spotřebičům daří i na úrovni technologické. Poznání, že všechny spotřebiče jsou vlastně kuličkovými počítači, k tomu pomohlo nemalou mírou. Stejně tak pomohly programy, které jste pro spotřebiče vytvořili.

V rámci vědeckého bádání se počítač, televize a mobilní telefon pokoušeli a stále pokoušejí proniknout do tajů kuličkového počítače. Zrovna nedávno se jim podařil velmi významný objev – zjistili, že v rámci programů pro kuličkové počítače mohou existovat i podprogramy!

Zavedení podprogramu

Podprogram je pojmenovaný úsek kódu (v našem případě tedy diagramu), který může být opakovaně volán z jiných míst programu. Spolu s voláním podprogramu lze specifikovat parametry – tj. kyblíčky, se kterými má podprogram pracovat.

Zápis podprogramu se nijak neliší od zápisu programu, kromě toho, že k podprogramu musí být pro identifikaci přiřazeno jeho jméno a počet parametrů, které má na vstupu. Pro přehlednost jméno podprogramu uvozujeme závorkami `()`, tj. např `Soucet()`. V každém podprogramu taktéž existují speciální bloky pro začátek a konec podprogramu, ty v tomto případě značíme jako `SP` a `EP`. Jediný rozdíl oproti blokům `S` a `E` je v tom, že blok `EP` značí návrat k vykonávání kódu do místa, odkud byl podprogram zavolán a nikoliv konec celého programu. Zároveň, vzhledem k čitelnosti, je vhodné, aby u každého podprogramu byl okomentován význam vstupních parametrů a funkcionalita daného podprogramu (viz popis u obrázku níže).

Volání podprogramu (tj. začátek vykonávání instrukcí podprogramu) se provede stejně jako jakákoliv jiná operace s kyblíčky, jen je zapotřebí napsat jméno podprogramu a v závorce specifikací parametrů, které jsou mu předávány. Uvnitř podprogramu se režim práce s kyblíčky liší oproti operacím mimo podprogram, a to konkrétně takto:

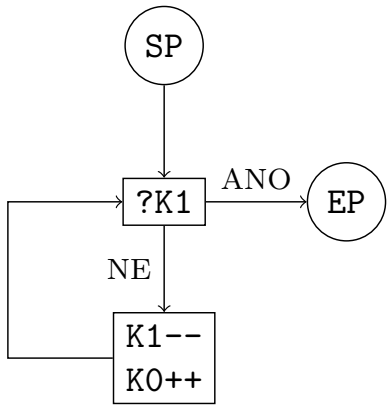
1) Má-li podprogram k parametrů (a tím pádem se mu předává k kyblíčků), jsou pro práci uvnitř podprogramu předané kyblíčky přemapovány na čísla kyblíčků $0, 1, \ldots, k-1$. Tj. předané kyblíčky jsou uvnitř podprogramu přístupné jako kyblíčky $K_0, K_1, \ldots, K_{k-1}$. Přístup k takovémuto kyblíčku stále ovlivňuje obsah kyblíčku, který je takto přemapován. To znamená, že ono přemapování je pouze jakýsi odkaz na původní kyblíček. Umístěním kuliček do těchto kyblíčku tak lze např. předávat výsledky z podprogramu zpět do programu.
2) Ostatní kyblíčky přístupné podprogramu mají čísla $k, k+1, \ldots$. Takto adresované kyblíčky taktéž neodpovídají originálním registrům kuličkového počítače, jedná se opět o odkazy

do nějakých fyzických kyblíčků. Do jakých přesně není třeba zkoumat, neboť je zaručeno, že tyto kyblíčky vybere pomyslný kompilátor sám tak, aby nijak neovlivňovaly obsah již využitých kyblíčků. Stejně tak je zaručeno, že obsah těchto kyblíčků bude jak na začátku vykonávání podprogramu, tak po jeho ukončení, nulový (tj. budou obsahovat nula kuliček). Volání podprogramu tak může ovlivnit počet kuliček pouze v těch kyblíčcích, které jsou mu přímo předány jako parametry.

Uvnitř podprogramu lze volat další podprogramy. Ale vzhledem k tomu, že model kuličkového počítače je konečný, není povoleno uvnitř podprogramu znovu volat stejný podprogram, a to ani nepřímo přes jiné podprogramy! Podprogramy jsou tak fakticky jen nástroj, jak šetřit tužku a papír. Každé volání podprogramu lze totiž za daných podmínek brát pouze jako vložení kódu podprogramu do jeho nadřízeného programu (plus zpětné přemapování kyblíčků a vynulování určitých kyblíčků používaných v podprogramu).

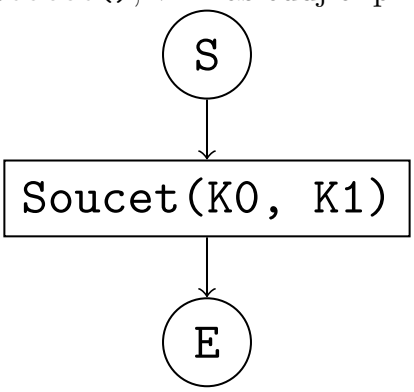
Jako příklad uvedeme opět operaci sčítání, tentokrát jako podprogram.

`Soucet()`, 2 parametry:



Komentář: Podprogram bere jako parametry dva kyblíčky K_0 a K_1, v každém z nich je zadáno libovolné číslo. Jako výsledek podprogramu se v K_0 vrátí součet daných čísel.

Pro vytvoření programu pro součet dvou čísel, která jsou na začátku zadaná jako počty kuliček v kyblíčcích K_0 a K_1 a výsledek je na konci programu umístěn v kyblíčku K_0, tak nyní stačí zavolat podprogram `Soucet()`, viz následující příklad:



Dalším velmi pěkným výsledkem, ke kterému chytré spotřebiče došly, je binární reprezentace čísel v kuličkovém počítači. Obsahuje-li tak kyblíček a kuliček, lze na tyto kuličky pohlížet také ve dvojkovém zápise, tj. jako na $(a)_2$. Konkrétně tak na číslo 14 lze nahlížet jako na tuto posloupnost bitů: $(14)_2 = 1110$. S určitými operacemi nad binárními zápisy čísel si však spotřebiče nedokáží poradit. Pomůžeš jim sestrojením následujících podprogramů? Věz, že v rámci řešení můžeš využívat již naprogramované operace z minula, a to jako podprogramy (ačkoliv jsme je specifikovali jako programy). Konkrétně jde o operace sčítání, násobení, celočíselné dělení a zbytek po celočíselném dělení. Samozřejmostí je, že si můžeš vytvořit libovolné další podprogramy dle svého uvážení a také využívat libovolné podprogramy ze svého řešení (není třeba jejich zápis znovu opakovat).

Soutěžní úlohy

a) Podprogram pro výpočet určité mocniny čísla 2

- Vstup: Jedno celé číslo a reprezentované počtem kuliček v kyblíčku K_0.
- Výstup: Číslo v kyblíčku K_0, které značí výsledek operace 2^a.

b) Podprogram pro výpočet bitového posunu vlevo/vpravo

- Vstup: Tři celá čísla a, b, c reprezentovaná počtem kuliček v kyblíčcích K_0, K_1 a K_2. Platí, že $c \in \{1, 2\}$.
- Výstup: Číslo v kyblíčku K_0, které vznikne bitovým posunem čísla a o b bitů směrem daným číslem c: pro $c = 1$ doleva, pro $c = 2$ doprava.
- Příklad: Operace bitový posun značí posunutí bitů daného čísla o daný počet pozic určitým směrem. U bitového posunu vlevo nově vzniklé pozice bitů doplňujeme nulami. U bitového posunu vpravo se bity, jejichž pozice zanikne, prostě zahodí. Konkrétně např. bitový posun čísla 5 o dva bity vlevo vyrobí z čísla $(5)_2 = 101$ číslo 10100. Naopak, bitový posun stejného čísla o jednu pozici doprava vyrobí číslo 10.

c) Podprogram pro výpočet bitové operace AND

- Vstup: Dvě celá čísla a, b reprezentovaná počtem kuliček v kyblíčcích K_0, K_1
- Výstup: Výsledek bitové operace $a\&b$ v kyblíčku K_0.
- Příklad: Bitová operace $a\&b$ porovnává odpovídající bity stejných řádů v binárních zápisech čísel a a b. Bit v jednom řádu bude ve výsledném čísle nastaven na 1, právě tehdy, když oba dva příslušné bity jsou jedničkové v čísle a i b. Tedy např $5\&12 = 101\&1100 = 100$.

d) Podprogram pro ověření palindromicity binárního zápisu čísla

- Vstup: Celé číslo a reprezentované počtem kuliček v kyblíčku K_0.
- Výstup: Jedna kulička v kyblíčku K_0, je-li $(a)_2$ palindromem (tj. čte-li se binární zápis čísla a zepředu stejně jako zezadu), jinak žádná kulička v kyblíčku K_0.
- Příklad: Číslo $a = 9$ je v binárním zápise palindromem, jelikož $(a)_2 = 1001$. Naopak, číslo $a' = 6$ palindromem v binárním zápise není, jelikož $(a')_2 = 110$.

Navrhni tyto podprogramy!