Zadanie č. 1

- Naprogramujte program, ktorý interpretuje zdrojový kód programovacieho jazyka popísaného nižšie. Váš program tento zdrojový kód vykoná (interpretuje) v súlade s platnými inštrukciami jazyka. Ak zdrojový kód obsahuje syntaktické chyby, je potrebné o tom minimálne vypísať **chybovú hlášku**!
- V súlade s popisom programovacieho jazyka a jeho fungovania je potrebné zabezpečiť pre daný program vstup a výstup.
- Vstupom Vášho programu, bude teda:
  - Zdrojový kód, ktorý sa má interpretovať, v textovej podobe.
  - Pole bajtov predstavujúce vstup interpretovaného programu.
- Výstupom Vášho programu bude:
  - Pole bajtov predstavujúce výstup interpretovaného programu.
- Súbor so zdrojovým kódom, súbor so vstupnými dátami a súbor s výstupnými dátami budú tvoriť jednotlivé argumenty pri spúšťaní interpretera **v tomto poradí**, t.j. ak má váš program (implementácia interpretera) názov `interpreter.exe`, zdrojový kód `source.txt`, súbor so vstupnými dátami `input.bin` a súbor s výstupnými dátami `output.bin` potom spustenie programu bude mať tvar

  ```
  interpreter.exe source.txt input.bin output.bin
  ```

  Prosím o dodržanie tejto konvencie - t.j. že interpretovaný program je v textovom súbore a tvorí prvý argument programu, vstup interpretovaného programu je v binárnom súbore a tvorí druhý argument programu a výstup interpretovaného programu bude uložený do binárneho súboru, ktorý predstavuje tretí argument programu.
- Deadline zadania je 19.03.2017, 23:59:59, t.j. polnoc z nedele na pondelok. Do AIS odovzdajte **zdrojový kód** Vášho programu (interpretera).
- Za naprogramovanie interpretera je 10 bodov.
- Zadanie môžete naprogramovať v jednom z jazykov: C, C++, C#, Java, Python, Go.

Je daný nasledujúci programovací jazyk:

- Programovací jazyk obsahuje 13 rôznych inštrukcií, ktoré sú kódované do binárnych štvoríc.
- Interpretácia jazyka sa vykonáva sekvenčne (t.j. inštrukcia za inštrukciou).
- Keď interpreter spracuje všetky inštrukcie, program končí, t.j. koniec programu je hneď za poslednou inštrukciou.
- Akýkoľvek neznámy symbol predstavuje syntaktickú chybu - program o tom vypíše **chybovú správu a skončí**.
- Exekúcia programu sa vykonáva nad **poľom buniek**, z ktorých každá má veľkosť 1 bajt (t.j. nad poľom bajtov). Každá bunka teda môže obsahovať hodnoty od 0 po 255 (8 bitov). Uvažujme pole buniek o veľkosti 100 000 buniek (t.j. 100 000 bajtov). Na začiatku sú všetky bunky nastavené na hodnotu 0x00.
- Program má tzv. **dátový ukazovateľ** (dátový pointer), ktorý sa hýbe medzi týmito bunkami a označuje aktuálnu bunku, nad ktorou sa vykonávajú operácie (inštrukcie). Na začiatku programu tento ukazovateľ ukazuje na prvú bunku poľa (prvý bajt). Vie sa posúvať doľava alebo doprava. Ak sa posúva vľavo z prvej bunky, tak sa cyklicky dostane na poslednú bunku. A naopak, posun vpravo z poslednej bunky je posunom na prvú bunku.
- Program taktiež podporuje **vstup** a **výstup**. Oba sú reprezentované binárnymi súbormi. Zo vstupného súboru vie program pomocou príslušnej inštrukcie načítať 1 bajt. Taktiež vie, pomocou príslušnej inštrukcie, zapísať 1 bajt do výstupného súboru.
- Jedná o mierne poznemený programovací jazyk *Nameless language*, zo stránky: `https://esolangs.org/wiki/Nameless_language`.

Tabuľka príkazov programu:

0000 posun dátového pointera vpravo o 1 pozíciu
0001 posun dátového pointera vľavo o 1 pozíciu
0010 inkrementuj (zvýš o 1) hodnotu bajtu na pozícii, kde ukazuje dátový pointer
0011 dekrementuj (zníž o 1) hodnotu bajtu na pozícii, kde ukazuje dátový pointer
0100 zapíš bajt do výstupu z pozície, kde ukazuje dátový pointer
0101 načítaj bajt zo vstupu do pozície, kde ukazuje dátový pointer
0110 skoč na ďalšiu inštrukciu za prislúchajúcou 0111,
ak je na pozícii, kde ukazuje dátový pointer, bajt 0x00
0111 skoč späť na inštrukciu za prislúchajúcu 0110,
ak je na pozícii, kde ukazuje dátový pointer, nenulový bajt (t.j. nie 0x00)
1000 Pripočítaj dekadickú reprezentáciu nasledovnej inštrukcie
ku hodnote bajtu na pozícii, kde ukazuje dátový pointer.
1001 Odpočítaj dekadickú reprezentáciu nasledovnej inštrukcie
od hodnoty bajtu na pozícii, kde ukazuje dátový pointer.
1010 Dummy inštrukcia (NOP), t.j. nič sa nedeje
1011 Vynuluj hodnotu bajtu, na ktorý ukazuje dátový pointer, t.j. nastav ho na 0x00.
1100 Nastav dátový pointer na začiatok pamäte, t.j. na prvú bunku s indexom 0.

Príkazy 0110 a 0111 sú párové, t.j. ku každému 0110 musí existovať 0111 a naopak. De facto predstavujú cyklus. Ak sa nepodarí nájsť pár, t.j. 0111 ku 0110 alebo 0110 ku 0111, ide o **syntaktickú chybu**.

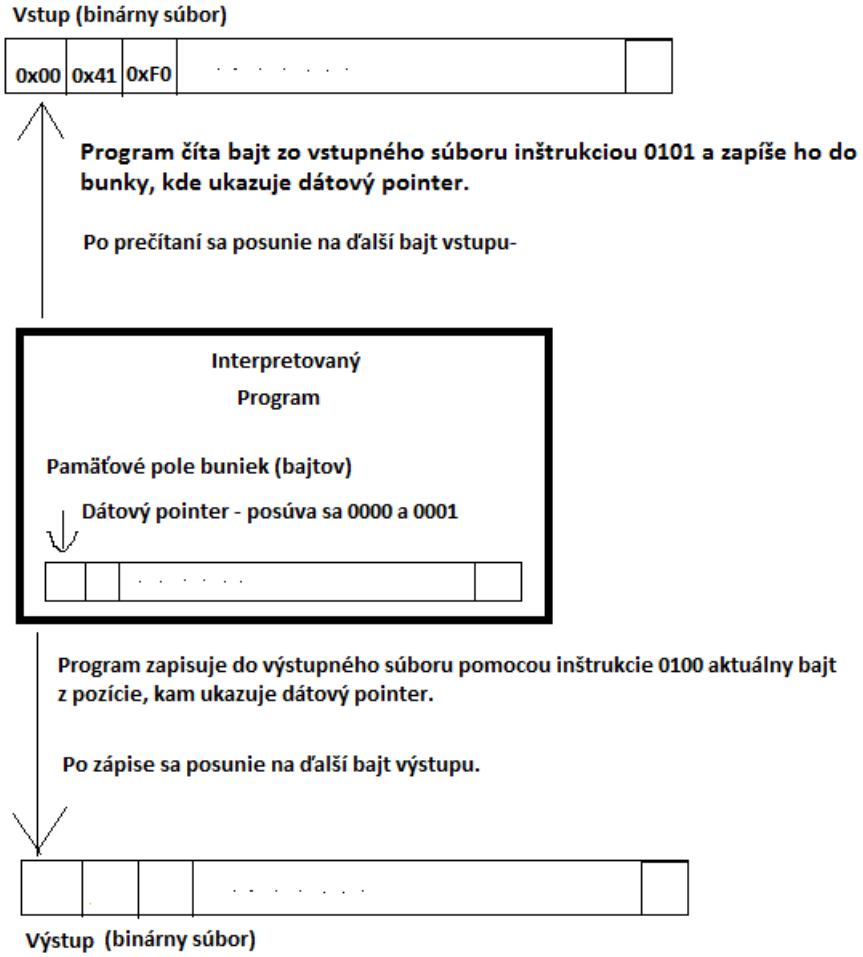
Iný typ **syntaktickej chyby** môže nastať, ak za inštrukciou 1000 alebo 1001 nenasleduje iná inštrukcia.

Nezabudnite na to, že príkazy 0110 a 0111 **môžu byť aj vnorené**!!! Dôležitou súčasťou je teda správne spárovanie týchto inštrukcií.

V prípade, že program chce čítať zo vstupu dáta, t.j. vykoná sa inštrukcia 0101, ale na vstupe už žiadne dáta nie sú, program **vypíše chybovú hlášku** a skončí.

Inkrementácia hodnoty 0xFF spôsobí jej zmenu na 0x00 (netreba to teda brať ako chybu, že sa nemá *kam inkrementovať*, ale ako normálny overflow). Taktiež dekrementácia 0x00 spôsobí zmenu na 0xFF.

Výpočtový model interpretovaného jazyka je nasledovný:



Obr. 1: Výpočtový model interpretovaného jazyka

Príklady programov pre lepšiu ilustráciu:
1) Program, ktorý načíta 1 bajt zo vstupu, inkrementuje ho o 1 a zapíše na výstup:

```
010100100100
```

Vysvetlenie: 0101 načíta 1 bajt zo vstupu do pozície, kam ukazuje dátový pointer (t.j. na začiatku je to prvá bunka). 0010 inkrementuje hodnotu, kam ukazuje dátový pointer (t.j. kam sa v predchádzajúcom kroku uložil bajt zo vstupu). 0100 následne túto novú inkrementovanú hodnotu pošle na výstup (znovu ako obsah bunky, t.j. 1 bajt). V prípade, že bol vstup povedzme bajt s hodnotou 0x41 (t.j. v ASCII písmeno A), tak sa na výstup zapíše bajt s hodnotou 0x42 (t.j. písmeno B).

2) Program, ktorý vypíše na výstup znak A (znak A má s ASCII tabuľke dekadickú hodnotu 65):

```
00100010001000100010001000100010
0110
000000100010001000100010001000100010
00010011
0111
000000100100
```

Program najprv nastaví prvú bunku na hodnotu 8. Inštrukcia 0110 znamená, že sa testuje, či je bunka, kam ukazuje dátový pointer (t.j. momentálne prvá) rovná nule. Ak by bola, program skočí za príslušnú opačnú zátvorku 0111. Keďže nie je, (lebo je rovná 8), program pokračuje s ďalšou inštrukciou. Tá hovorí, aby sa dátový pointer posunul vpravo o 1 pozíciu. Následne sa táto bunka 8-krát inkrementuje. V tomto momente sú teda v poli buniek prvé 2 bunky (prvé 2 bajty) rovné 0x08 a 0x08. Na ďalšom riadku je prvou inštrukciou posun dátového pointera vľavo (t.j. späť na prvú bunku) a jej dekrementácia o 1. Teraz sú prvé 2 bunky rovné 0x07 a 0x08. Nasleduje inštrukcia 0111, ktorá otestuje, či je aktuálna bunka rovná nule. Keďže aktuálne je dátový pointer na prvej bunke a tam je bajt 0x07, tak sa program znovu vracia na inštrukciu za 0110. Takto vlastne docielime, že vždy, keď sa prvý bajt dekrementuje o 1, tak druhý bajt sa inkrementuje o 8. Po 8-iteráciách bude prvý bajt 0x00 a druhý bajt 0x41 (dekadicky 65). Keď inštrukcia 0001 dekrementuje prvý bajt a ten dosiahne hodnotu nula, ďalšou je znovu inštrukcia 0111. Tá testuje, či je bajt, na ktorý ukazuje dátový pointer, nula.

Keďže je, tak sa už nevracia znovu na inštrukciu za 0110, ale pokračuje ďalej, t.j. vyjde zo slučky von na ďalšie inštrukcie 000001000100. T.j. posunie sa znovu na 2. bunku, inkrementuje ju o 1 (čím má teraz hodnotu 0x41) a vypíše ju na výstup. Keďže bajt s hodnotou 0x41 je znak A, program de facto na výstup vypíše znak A.

3) Program, ktorý vypíše na výstup znak A, verzia B:

```
001000100010001000100010
0110
0000
10001010
0001
0011
0111
0000
00100010001000100010
0100
```

Program na začiatku nastaví prvú bunku na hodnotu 6 (0x06). Potom vojde do cyklu, v ktorom sa posunie na druhú bunku a následne k nej pripočíta pomocou inštrukcie 1000 dekadickú hodnotu nasledovnej inštrukcie 1010, t.j. číslo 10. Po pripočítaní vykoná inštrukciu 1010, ale keďže je to NOP, nič sa neudeje. Potom sa posunie dátový pointer naspäť na prvú bunku a dekrementuje ju o jedna. Nasleduje koniec cyklu, t.j. 0111 a kontrola, či je aktuálny bajt nulový. Ak nie je, vracia sa na začiatok cyklu, ak je, pokračuje ďalej tým, že sa znovu posunie na druhú bunku (ktorá má momentálne hodnotu 60, t.j. 0x3C) a 5-krát ju zvýši o jedna, čím sa jej hodnota nastaví na 65 (0x41, t.j. A) a zapíše na výstup.

4) Program, ktorý načíta 5 bajtov zo vstupu, každý bajt uloží do inej pamäťovej bunky a zvýši ho o jedna. Následne sa vráti na začiatok pamäte a bajty zapíše na výstup.

```
010100100000
010100100000
010100100000
010100100000
01010010
1100
01000000
01000000
01000000
01000000
01000000
```

5) (Holy Grail) Program na výstup vypíše (uloží bajty predstavujúce) reťazec **Hello World!**. Všimnite si (hoci to možno hneď nie je vidno), že obsahuje cyklus v cykle.

```
00100010001000100010001000100010
0110
00000010001000100010
0110
000000100010
0000001000100010
0000001000100010
00000010
00010001000100010011
0111
00000010
00000010
00000011
000000000010
011000010111
00010011
0111
000000000100
00000011001100110100
00100010001000100010001000100100010000100010001000100100
000000000100
000100110100
00010100
0010001000100100
00110011001100110011001100110100
00110011001100110011001100110011001100110100
0000000000100100
```