

Zadanie číslo 4 – Jukapo RELOADED powered by Ohybný bizón

Mladý talentovaný fínsky formálny lingvista so slovenskými koreňmi Wilhelmus Kasa von Verešvár si po študovaní formálnych jazykov a automatov uvedomil, že vie napísať jednoduchý prekladač svojho programovacieho jazyka Jukapo prostredníctvom programov Bison a Flex. Vašou úlohou je mu pomôcť a napísať **jazykový procesor**, ktorý preloží program napísaný v jazyku Jukapo do **medzikódového zápisu pomocou upraveného jazyka štvoríc**.

Vašou úlohou je použiť nástroje **Bison a Flex** na zostrojenie jazykového procesora, ktorý spracuje program zapísaný v jazyku Jukapo a ktorého výstupom je postupnosť inštrukcií v jazyku štvoríc sémanticky ekvivalentná programu v jazyku Jukapo.

Váš jazykový procesor musí zároveň oznámiť, či sa pri preklade vyskytla:

- lexikálna chyba
- syntaktická chyba
- sémantická chyba

Pre jednoduchosť uvažujte, že vstup do jazykového procesora (program v jazyku JUKAPO) môže byť uložený v súbore a do jazykového procesora sa presmeruje pri jeho spúšťaní (viď. Príklady).

Výstup procesora (štvorice / chybové správy) nech je na štandardný výstup (obrazovku).

-
- Za korektne vypracované zadanie je 10 bodov.
 - Zadanie vypracujte **použitím nástrojov Flex a Bison**
 - Deadline zadania je 20.5.2020, 23:59. **ŽIADNE POSÚVANIE TERMÍNU NIE JE MOŽNÉ!**
 - Do AIS odovzdajte súbory, ktoré predstavujú vstupy do programov Bison a Flex (t.j. štandardne sa označujú ako súbory s koncovkami **.y** a **.l** podľa prednášok). Taktiež, ak máte naprogramované pomocné funkcie v ďalších súboroch, odovzdajte aj tie!
 - Pripomínam, že Flex a Bison pracujú s jazykom C.
 - A ako vždy, zistené podvádzanie pri vypracúvaní zadania bude hodnotené zaFXovaním známky v AIS.

Jazyk JUKAPO

Pre pripomenutie, jazyk JUKAPO pozostáva z týchto inštrukcií:

premenna id	Vytvorí premennú <i>id</i>
nacitaj id	Program načíta obsah premennej <i>id</i> z klávesnice. Na obrazovku vypíše info o tom, aká premenná sa načítava.
vypis id	Program vypíše obsah premennej <i>id</i> na obrazovku. Na obrazovku vypíše info o tom, aká premenná sa vypisuje.
id1 = id2 + id3	Do premennej <i>id1</i> uloží súčet <i>id2 + id3</i>
id1 = id2 - id3	Do premennej <i>id1</i> uloží rozdiel <i>id2 - id3</i>
id1 = id2 * id3	Do premennej <i>id1</i> uloží súčin <i>id2 * id3</i>
id1 = id2	Do premennej <i>id1</i> vloží hodnotu <i>id2</i>
opakuj id1 < id2	Ak platí, že <i>id1 < id2</i> , začni nasledujúce príkazy vykonávať v cykle.
opakuj id1 > id2	Ak platí, že <i>id1 > id2</i> , začni nasledujúce príkazy vykonávať v cykle.
opakuj id1 <= id2	Ak platí, že <i>id1 <= id2</i> , začni nasledujúce príkazy vykonávať v cykle.
opakuj id1 >= id2	Ak platí, že <i>id1 >= id2</i> , začni nasledujúce príkazy vykonávať v cykle.
opakuj id1 == id2	Ak platí, že <i>id1</i> a <i>id2</i> sú rovnaké, začni nasledujúce príkazy vykonávať v cykle.
opakuj id1 != id2	Ak platí, že <i>id1</i> a <i>id2</i> sú rôzne, začni nasledujúce príkazy vykonávať v cykle.
jukapo	Koniec cyklu.

Vstupný súbor teda bude pozostávať z týchto inštrukcií.

Oproti zadaniu č. 1 je tu explicitná zmena – program v jazyku JUKAPO bude NAJPRV obsahovať deklarácie premenných a až potom budú nasledovať ostatné príkazy!

Pre lepšiu predstavu o syntaxe jazyka JUKAPO uvádza lingvista Wilhelmus Kasa formálnu gramatiku popisujúcu jeho syntax:

Syntax jazyka JUKAPO (jedná sa o SLR(1) gramatiku, NIE JE to LL(1) gramatika)

<program> → <deklaracie> <prikazy>
<deklaracie> → <deklaracia> <deklaracie>
<deklaracie> → ε
<deklaracia> → **premenna id**
<prikazy> → <prikaz> <prikazy>
<prikazy> → ε
<prikaz> → **nacitaj id**
<prikaz> → **vypis id**
<prikaz> → <priradenie>
<prikaz> → **opakuj** <podmienka> <prikazy> **jukapo**
<priradenie> → **id** = <vyraz>
<vyraz> → <hodnota> <operator> <hodnota>
<vyraz> → <hodnota>
<operator> → +
<operator> → -
<operator> → *
<podmienka> → <hodnota> <komparator> <hodnota>
<hodnota> → **id**
<hodnota> → **konst**
<komparator> → >
<komparator> → <
<komparator> → >=
<komparator> → <=
<komparator> → ==
<komparator> → !=

Neterminálne symboly: <program>, <deklaracie>, <deklaracia>, <prikazy>, <prikaz>, <priradenie>, <vyraz>, <hodnota>, <operator>, <podmienka>, <komparator>

Počiatočný neterminál: <program>

Terminálne symboly (lexémy): **premenna, id, nacitaj, vypis, opakuj, jukapo, konst, +, -, *, =, ==, !=, >=, <=, >, <**

Lexémy **premenna, nacitaj, vypis, opakuj, jukapo, +, -, *, =, ==, !=, >=, <=, >, <** budú rozpoznané vo vstupnom súbore ako reťazce **premenna, nacitaj, vypis, opakuj, jukapo, +, -, *, =, ==, !=, >=, <=, >, <**

Lexéma **id** predstavuje reťazec začínajúci malým/veľkým písmenom anglickej abecedy za ktorým je reťazec malých písmen/veľkých písmen/číslíc, t.j. reťazec spĺňajúci regulárny výraz:

(a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z|A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z)(a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z|A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z|0|1|2|3|4|5|6|7|8|9)*

T.j. neformálne popísané ako (a|b|...|z|A|B|...|Z)(a|b|...|z|A|B|...|Z|0|1|...|9)*

Lexéma **konst** predstavuje číselnú konštantu **s možným záporným znamienkom na začiatku**, t.j. (- | ε)(0|1|2|3|4|5|6|7|8|9) | ((1|2|3|4|5|6|7|8|9)(1|2|3|4|5|6|7|8|9|0)*))

(t.j. alebo jednociferné čísla, alebo viacciferné čísla bez núl na začiatku, obe potenciálne záporné)

Inštrukcie upraveného jazyka štvoríc

Inštrukcie jazyka štvoríc, ktoré sa môžu vyskytnúť v preklade (jedná sa o mierne upravený jazyk štvoríc z prednášky)

Inštrukcia

(operátor, op1, op2, výsledok)

(-, op1, výsledok)

(:=, op1, op2, výsledok)

(INTEGER, op1, výsledok)

(JUMP, návěstie)

(JUMPT, op1, návěstie)

(JUMPF, op1, návěstie)

(READ, op1)

(WRITE, op1)

(LABEL, návěstie)

Význam inštrukcie

Priradí do *výsledok* \leftarrow *op1* operátor *op2*, pričom *operátor* $\in \{+, -, *, <, >, <=, >=, ==, !=\}$ (t.j. binárne aritmetické operátory plus, mínus, krát; binárne relačné operátory $<, >, <=, >=, ==, !=$)

výsledok \leftarrow -*op1* (unárne mínus)

výsledok \leftarrow *op1*, *op2* obsahuje počet prenesených bajtov (v našom prípade VŽDY sizeof(INTEGER)); inštrukcia priradenia

výsledok \leftarrow Integer(*op1*) (využíva sa pri načítaní číselných konštánt, konverzia na typ Integer)

Nepodmienený skok na inštrukciu s návěstím *návěstie*

Podmienený skok na inštrukciu s návěstím

návěstie, ak *op1* obsahuje logickú hodnotu TRUE

Podmienený skok na inštrukciu s návěstím *návěstie*, ak *op1* obsahuje logickú hodnotu FALSE

Načíta z klávesnice hodnotu a vloží do *op1*

Vypíše na obrazovku hodnotu z *op1*

Deklarácia návěstia *návěstie*, ktoré sa použije ako návěstie nasledujúcej inštrukcie

1. Výpis inštrukcií jazyka štvoríc bude obsahovať **každú inštrukciu na novom riadku**.
2. Nezabúdajte, že jazyk štvoríc používa tzv. dočasné premenné (temporaries) na ukladanie medzivýsledkov – uvažujte, že ich meno začína písmenom *t*, za ktorým sa nachádza číslo
3. Taktiež mená návěstí nech začínajú písmenom *n* a pokračujú číslom
4. Vhodne odlišujte mená návěstí a premenných, nech program funguje korektne!
5. Pri spracovaní konštánt vo vstupnom Jukapo súbore VŽDY najprv vložte danú konštantu do dočasnej premennej a následne pracujte s touto premennou!
6. Všetky inštrukcie z jazyka Jukapo sú v podstate veľmi ľahko prepísateľné na štvorice, s výnimkou cyklu! Tam sa musíte vy potrápiť s tým, ako zapíšete cyklus pomocou inštrukcií jazyka štvoríc! (Wilhelmus Kasa sa už teší na vaše výtvary...)

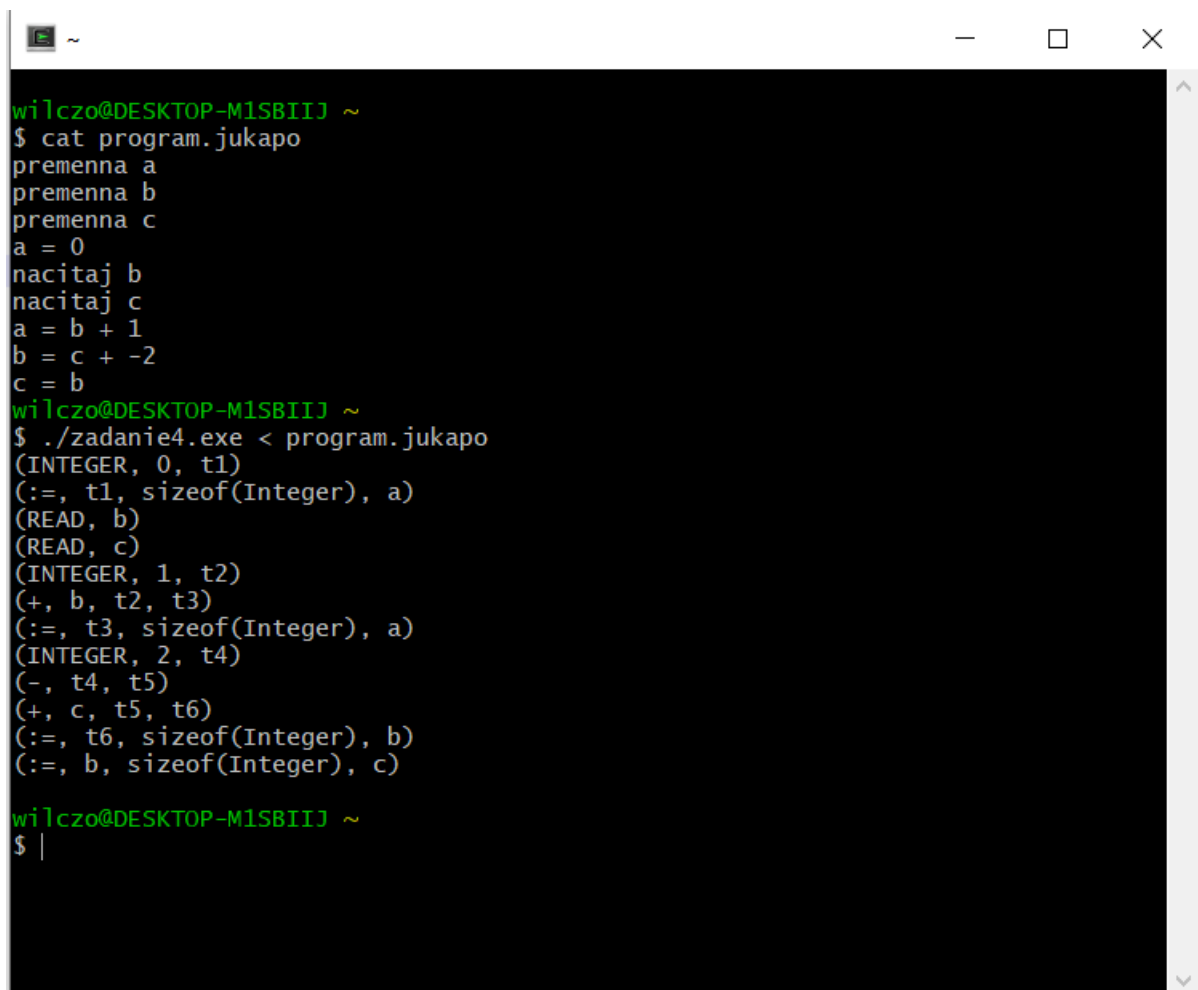
*Pozn.: V poslednej prednáške je malý komentár o inom, podobnom probléme - ako prepísať if <podmienka> then <príkaz> pomocou jazyka štvoríc – ale tam je len **pseudokód** ako to spraviť! Čiže reálne vo vašich riešeniach to musí byť postupnosť inštrukcií jazyka štvoríc z hore uvedenej tabuľky, ktorá to realizuje, nie len pseudokódová poznámka typu „Kód príkazu za then“.*

Chybové správy, ktoré musíte ošetriť:

1. **Lexikálna chyba** – program vypíše, že vstup obsahuje lexikálnu chybu, zároveň ukončí preklad a končí.
2. **Syntaktická chyba** – program vypíše, že vstup obsahuje syntaktickú chybu, zároveň ukončí preklad a končí.
3. **Sémantická chyba** – program vypíše, že vstup obsahuje sémantickú chybu, zároveň ukončí preklad a končí. Musíte ošetriť tieto sémantické chyby:
 1. redeklačia premennej – program vypíše, že bola redeklarovaná premenná aj s jej identifikátorom
 2. použitá nedeklarovaná premenná (pri príkazoch **nacitaj/vypis**, vo výrazoch, v podmienkach, pri priradení na ľavej strane) – program vypíše, že bola použitá premenná, ktorá nebola deklarovaná, aj s jej identifikátorom

Ukážka činnosti programu:

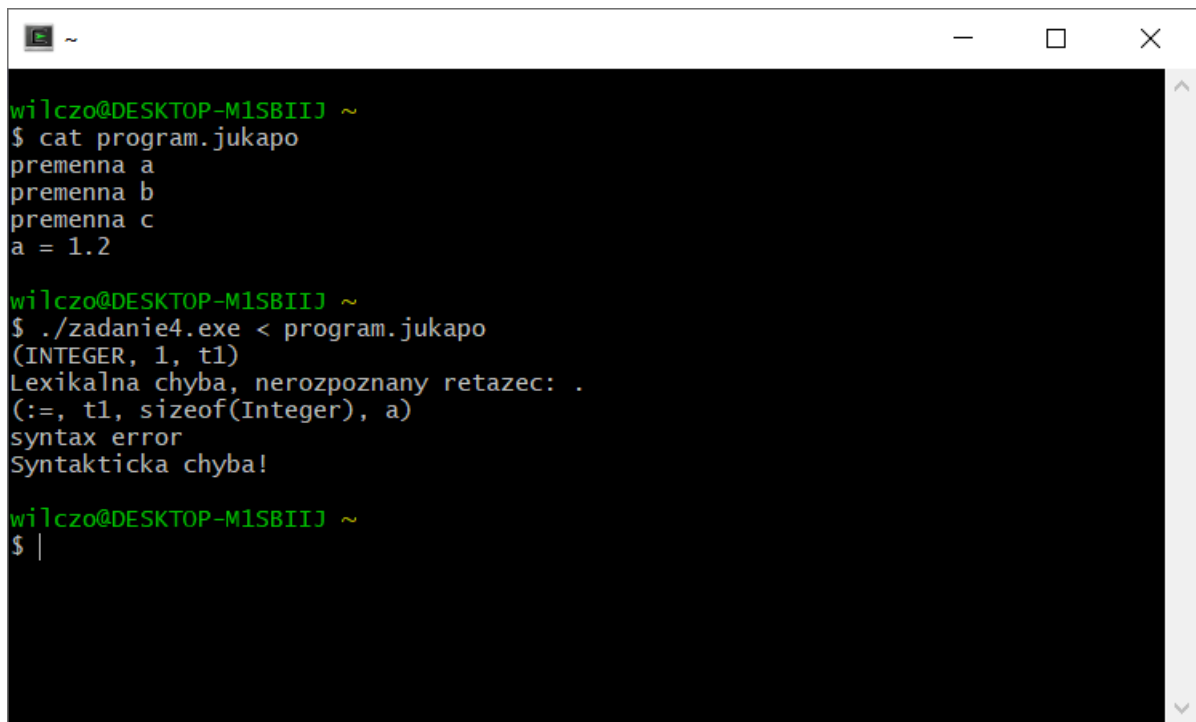
V súbore „program.jukapo“ je Jukapo program, v súbore „program.stvorice“ bude ekvivalentný program zapísaný v upravenom jazyku štvoríc:



```
wilczo@DESKTOP-M1SBIIIJ ~  
$ cat program.jukapo  
premenna a  
premenna b  
premenna c  
a = 0  
nacitaj b  
nacitaj c  
a = b + 1  
b = c + -2  
c = b  
wilczo@DESKTOP-M1SBIIIJ ~  
$ ./zadanie4.exe < program.jukapo  
(INTEGER, 0, t1)  
(:=, t1, sizeof(Integer), a)  
(READ, b)  
(READ, c)  
(INTEGER, 1, t2)  
(+, b, t2, t3)  
(:=, t3, sizeof(Integer), a)  
(INTEGER, 2, t4)  
(-, t4, t5)  
(+, c, t5, t6)  
(:=, t6, sizeof(Integer), b)  
(:=, b, sizeof(Integer), c)  
wilczo@DESKTOP-M1SBIIIJ ~  
$ |
```

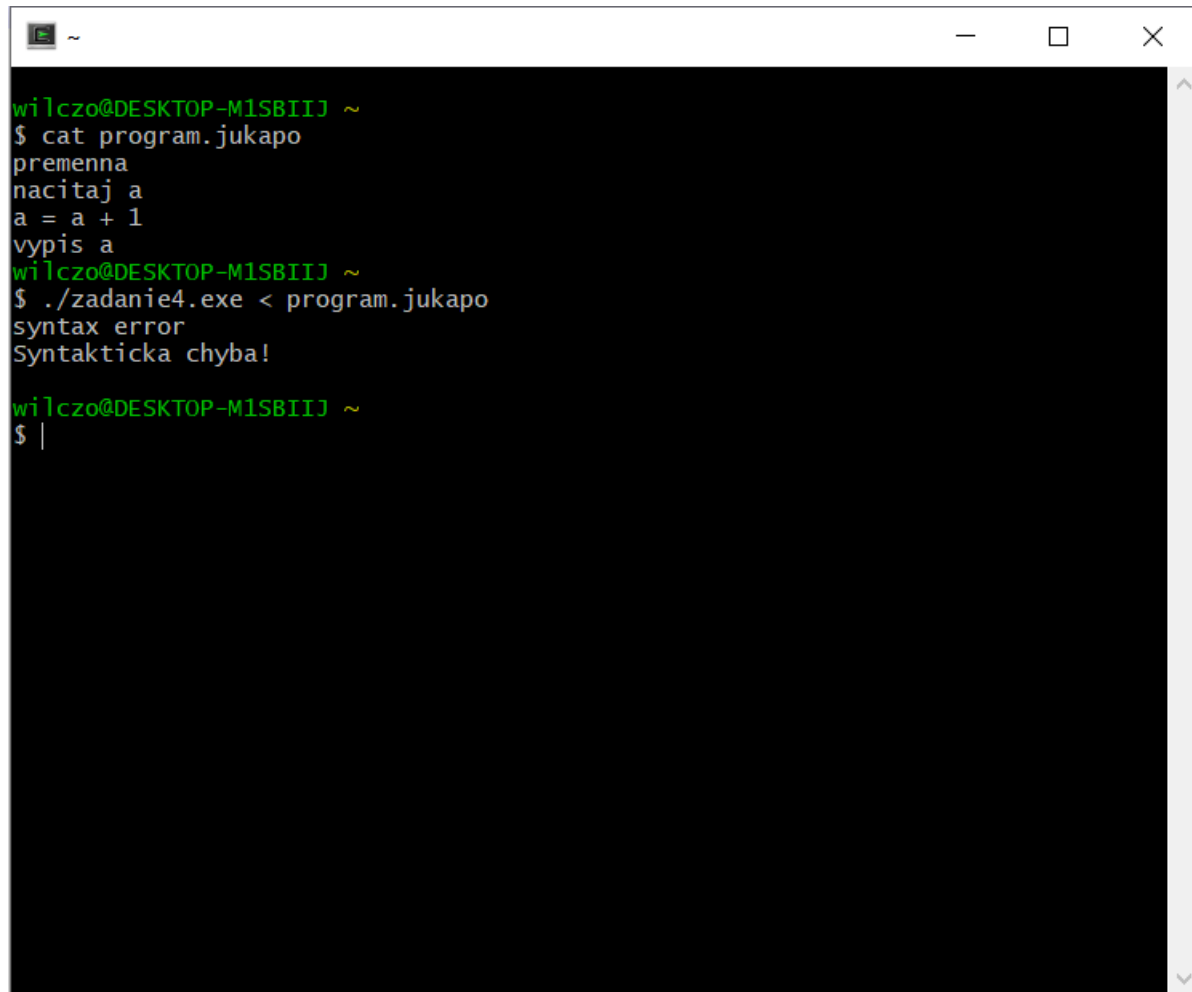
Tento program obsahuje lexikálnu chybu – netokenizovateľný reťazec na vstupe (bodka)

(výpis zároveň oznámi aj syntaktickú chybu, čo je korektné, keďže lexikálna chyba v podstate implikuje aj syntaktickú chybu)



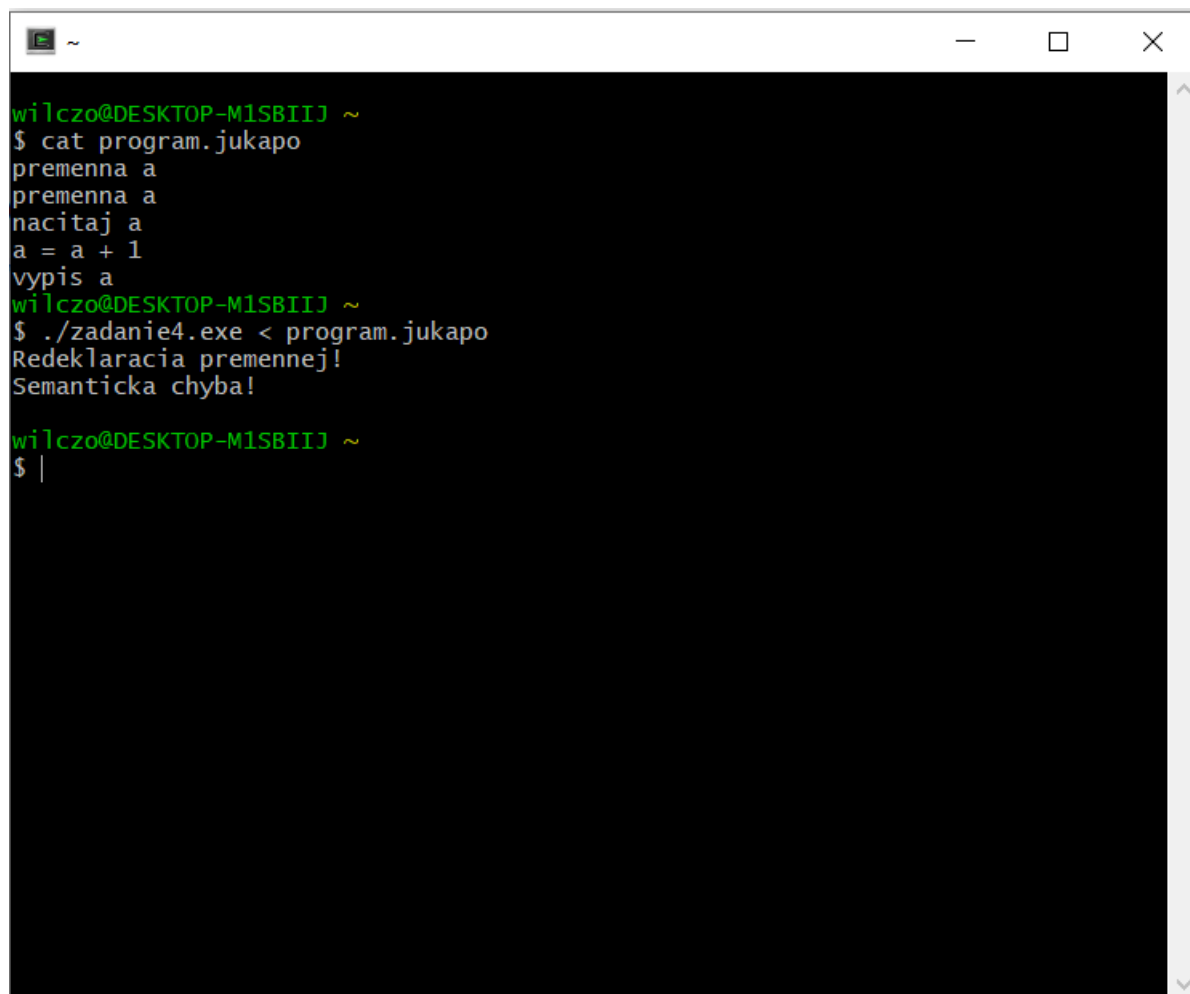
```
wilczo@DESKTOP-M1SBIIIJ ~  
$ cat program.jukapo  
premenna a  
premenna b  
premenna c  
a = 1.2  
  
wilczo@DESKTOP-M1SBIIIJ ~  
$ ./zadanie4.exe < program.jukapo  
(INTEGER, 1, t1)  
Lexikalna chyba, nerozpoznany retazec: .  
(:=, t1, sizeof(Integer), a)  
syntax error  
Syntakticka chyba!  
  
wilczo@DESKTOP-M1SBIIIJ ~  
$ |
```

Tento program obsahuje syntaktickú chybu – chýba meno premennej pri deklarácii



```
wilczo@DESKTOP-M1SBIIJ ~  
$ cat program.jukapo  
premenna  
nacistaj a  
a = a + 1  
vypis a  
wilczo@DESKTOP-M1SBIIJ ~  
$ ./zadanie4.exe < program.jukapo  
syntax error  
Syntakticka chyba!  
  
wilczo@DESKTOP-M1SBIIJ ~  
$ |
```

Tento program obsahuje sémantickú chybu – reдекlaráciu premennej *a*



```
wilczo@DESKTOP-M1SBIIJ ~  
$ cat program.jukapo  
premenna a  
premenna a  
nacitaj a  
a = a + 1  
vypis a  
wilczo@DESKTOP-M1SBIIJ ~  
$ ./zadanie4.exe < program.jukapo  
Redeklaracia premennej!  
Semanticka chyba!  
wilczo@DESKTOP-M1SBIIJ ~  
$ |
```