

# PROG 2

## Zadanie 3



# Obsah

- Odovzdávanie/podmienky
- Vysvetlenie a úlohy
- Bodovanie
- Živá ukážka
  - Nastavenie CMD argumentov v Clione
  - Použitie programu v termináli

# Odvzdávanie/podmienky

- **Deadline:** 2. 4. 2024, 09:59:59
- 10 bodov
- Anti-plagiátorský systém



[www.prog2.dev](http://www.prog2.dev)

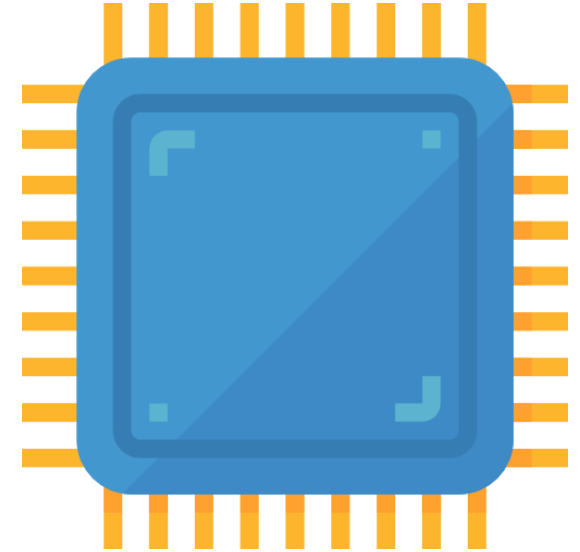
# Odovzdávanie/podmienky

## Penalizácia

- 1. pokus ..... **max. 10 b**
- 2. pokus ..... max. 9 b
- 3. pokus ..... max. 7 b
- 4. pokus ..... max. 5 b
- 5. pokus ..... max. 3 b
- 6. pokus ..... max. 1 b

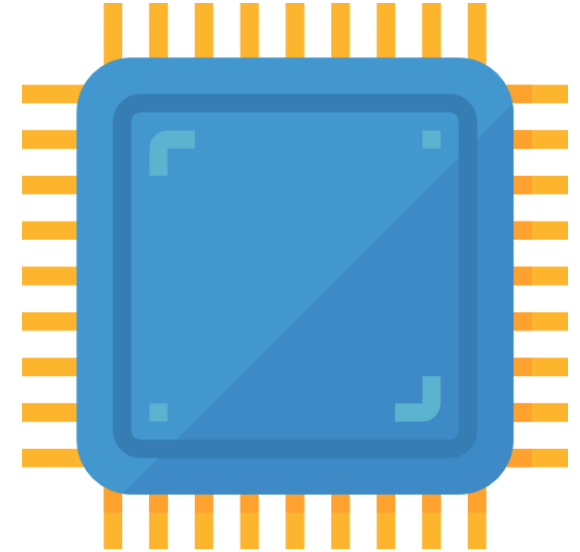
## Zadanie 3

- Cieľom zadania je naprogramovať jednoduchý konzolový textový procesor.



## Zadanie 3

- Cieľom zadania je naprogramovať jednoduchý konzolový textový procesor.
- **Hlavné úlohy:**
  - spracovanie command-line argumentov
  - načítanie textu
  - filtrovanie znakov
  - hľadanie a nahrádzanie slov



# Zadanie 3

V zadaní si **precvičíte**:

- znaky,
- reťazce,
- pointre,
- spracovanie command-line argumentov

# Command-line arguments

- Aplikácia bude riadená CMD (command-line) argumentmi, ktoré používateľ zadá pri spustení v termináli.





# Command-line arguments

- Aplikácia bude riadená CMD (command-line) argumentmi, ktoré používateľ zadá pri spustení v termináli.
- CMD argumenty slúžia na ovládanie správania sa programu z miesta jeho spustenia.



# Command-line argumenty

- Aplikácia bude riadená CMD (command-line) argumentmi, ktoré používateľ zadá pri spustení v termináli.
- CMD argumenty slúžia na ovládanie správania sa programu z miesta jeho spustenia.
- Každý CMD argument je reprezentovaný ako C reťazec.



# Command-line arguments

- Všetky CMD argumenty sú k dispozícii prostredníctvom poľa reťazcov *argv*, ktoré má dĺžku *argc* prvkov.



# Command-line arguments

- Všetky CMD argumenty sú k dispozícii prostredníctvom poľa reťazcov *argv*, ktoré má dĺžku *argc* prvkov.
- Prvok *argv[0]* obsahuje cestu k spúšťanému programu.



# Command-line arguments

- Všetky CMD argumenty sú k dispozícii prostredníctvom poľa reťazcov *argv*, ktoré má dĺžku *argc* prvkov.
- Prvok *argv[0]* obsahuje cestu k spúšťanému programu.
- Ostatné prvky poľa *argv* predstavujú používateľom zadané CMD argumenty.



# Command-line arguments

- Pole *argv* a jeho délka *argc* sú parametrami hlavnej funkcie main.

```
int main(int argc, char *argv[]) {  
    // your code  
    return 0;  
}
```

# Command-line argumenty

- Pole *argv* a jeho dĺžka *argc* sú parametrami hlavnej funkcie main.

```
int main(int argc, char *argv[]) {  
    // your code  
    return 0;  
}
```



???

# Command-line argumenty

- Pole *argv* a jeho dĺžka *argc* sú parametrami hlavnej funkcie main.

```
int main(int argc, char *argv[]) {  
    // your code  
    return 0;  
}
```

Technicky sa jedná o  
pole reťazcov



# Command-line arguments

- Príklad spustenia programu s CMD argumentmi v termináli.

```
z3.exe 123 abc "hello world"
```

# Command-line arguments

- Príklad spustenia programu s CMD argumentmi v termináli.

z3.exe 123 abc "hello world"

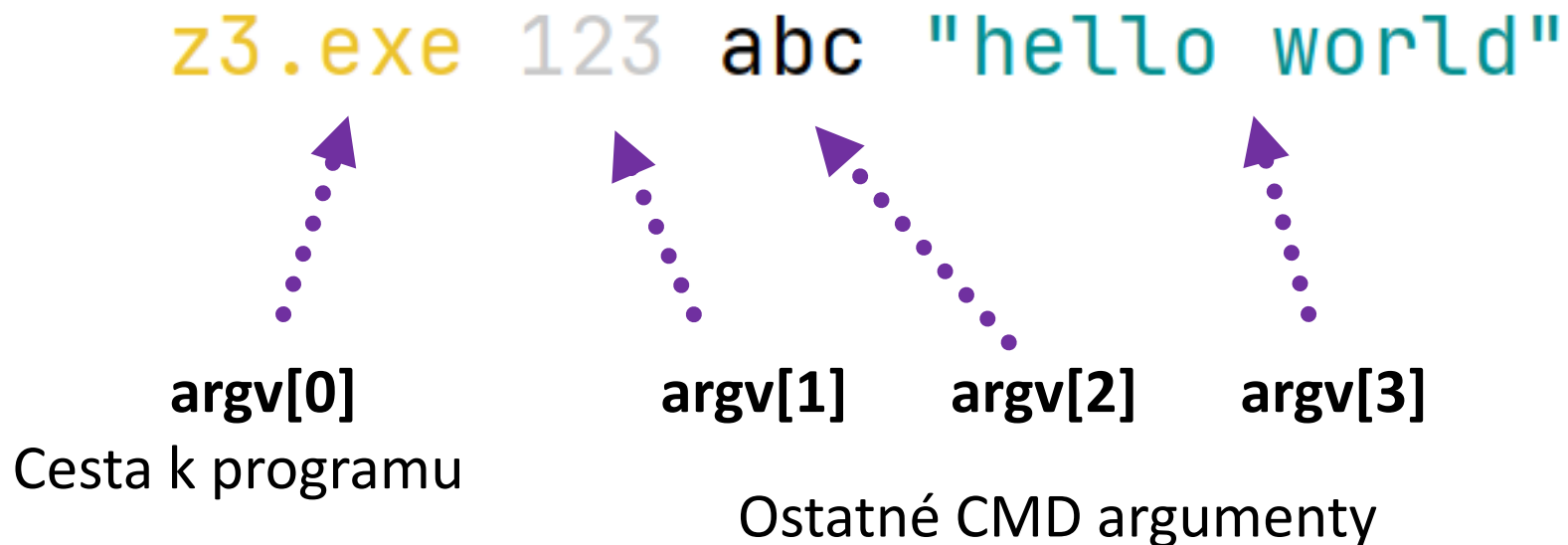


**argv[0]**

Cesta k programu

# Command-line arguments

- Príklad spustenia programu s CMD argumentmi v termináli.



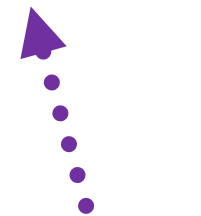
# Command-line arguments

- Príklad spustenia programu s CMD argumentmi v termináli.

```
z3.exe 123 abc "hello world"
```



**Pozor:** jedná sa o jeden viacslovný argument, nakoľko je uzavretý v úvodzovkách.

  
argv[3]

# Command-line argumenty

Budeme rozlišovať 3 typy CMD argumentov:

- Prepínače (z angl. options)
- Parametre prepínačov
- „Neprepínače“ (z angl. non-options)

# Prepínače (options)



- Prepínačom rozumieme CMD argument v tvare **-x**, kde **x** je ľubovoľné písmeno.
- Prepínače budú reprezentovať príslušné operácie spracovania načítaného textu.

# Parametre prepínačov



- Prepínač môže mať stanovený aj svoj parameter, napr. *-p abc*, kde za prepínačom *-p* nasleduje jeho parameter *abc*.

## „Neprepínače“ (non-options)

- Každý CMD argument, ktorý nepatrí ani do jednej z dvoch vyššie uvedených kategórií je považovaný za tzv. non-option argument.
- V prípade tohto zadania budú všetky non-option argumenty umiestnené až za všetkými prepínačmi a ich parametrami (t.j. budú zadané ako posledné).



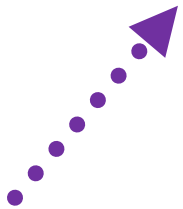
# CMD argumenty - príklad

```
z3.exe -a -b param arg1 arg2 arg3
```

# CMD argumenty - príklad

```
z3.exe -a -b param arg1 arg2 arg3
```

Cesta k  
programu

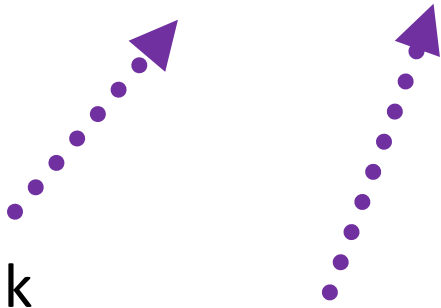


# CMD argumenty - príklad

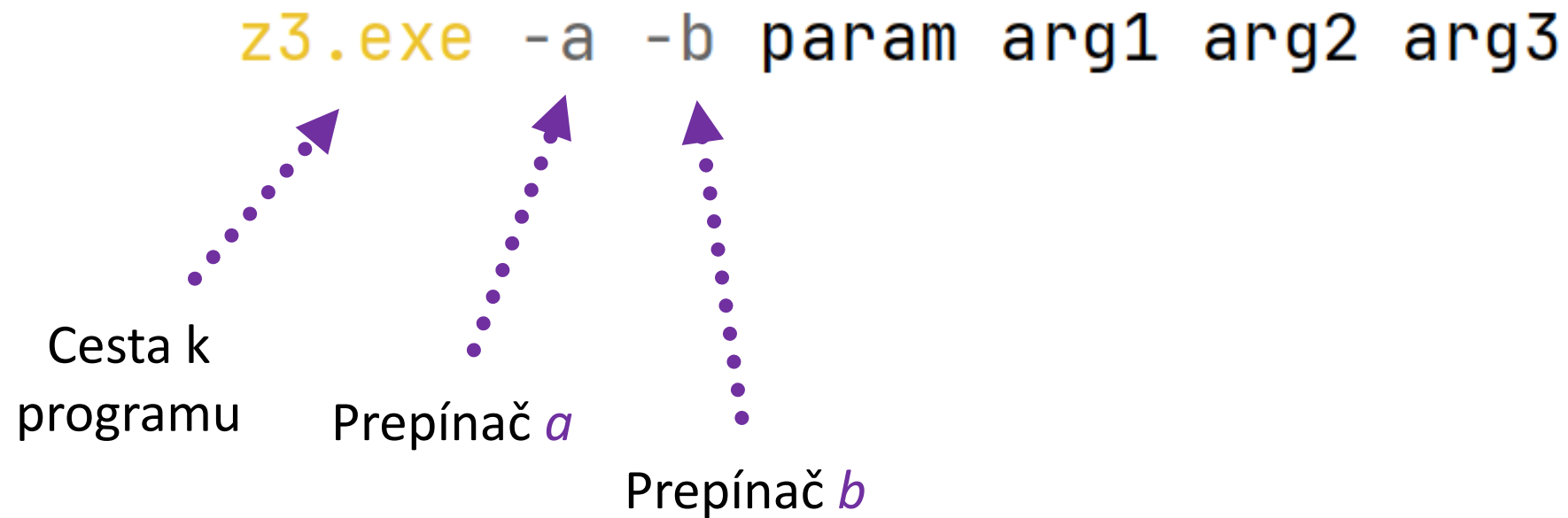
z3.exe -a -b param arg1 arg2 arg3

Cesta k  
programu

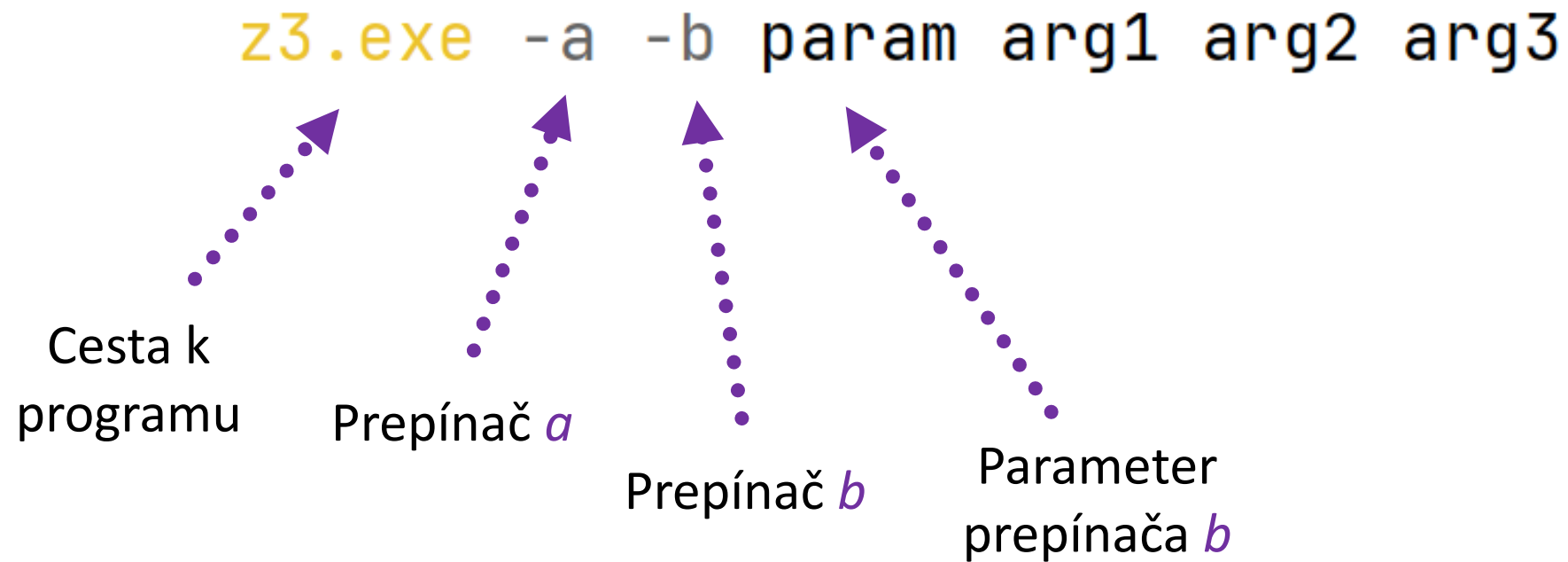
Prepínač *a*



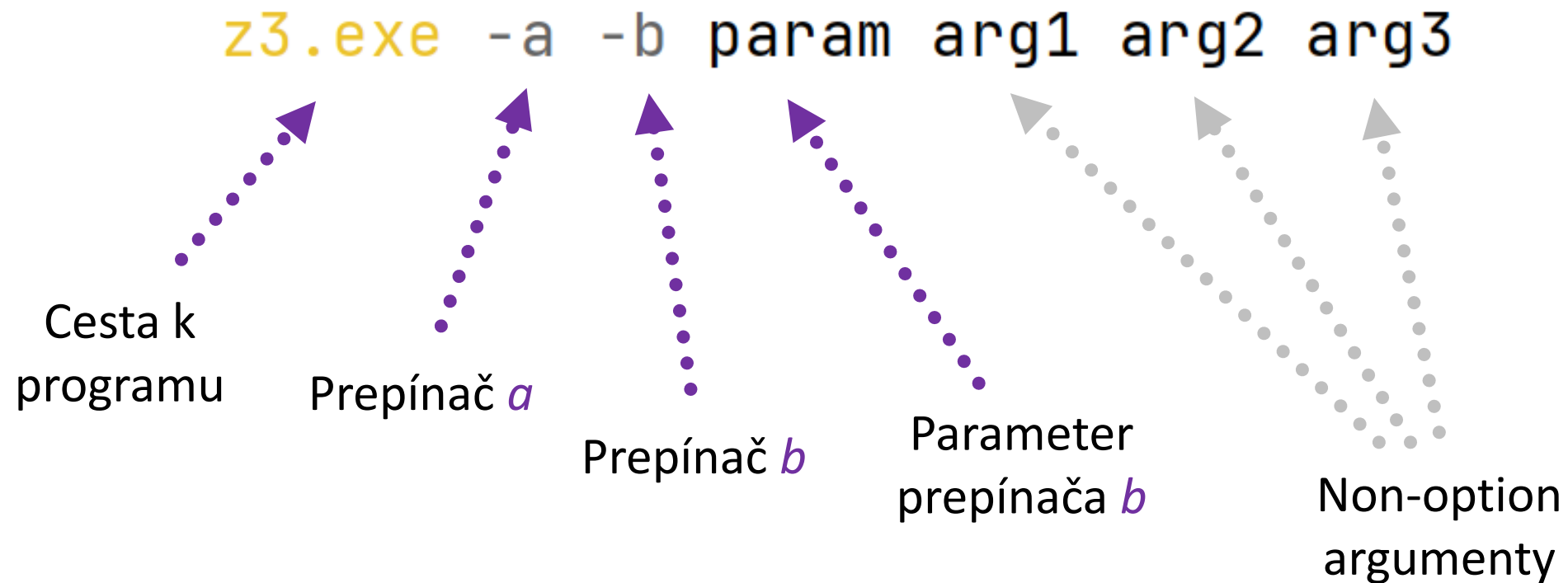
# CMD argumenty - príklad



# CMD argumenty - príklad



# CMD argumenty - príklad



# Funkcia *getopt*

- Proces rozpoznania CMD argumentov sa dá automatizovať pomocou knižničnej funkcie *getopt*.
- Funkcia *getopt* taktiež odhaľuje aj situácie nesprávneho použitia prepínačov.

# Funkcia getopt

- Táto funkcie nepatrí do štandardnej knižnice jazyka C, ale pre potreby tohto zadania ju môžete použiť.
- Je dostupná len v Linux prostredí (taktiež aj MinGW, Cygwin, MSYS2 a WSL).
- Treba do programu vložiť hlavičkový súbor [unistd.h](#)

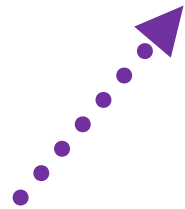


# Funkcia getopt

```
int getopt(int argc, char *argv[], const char *optstring);
```

# Funkcia getopt

```
int getopt(int argc, char *argv[], const char *optstring);
```




Počet CMD  
argumentov

# Funkcia getopt

```
int getopt(int argc, char *argv[], const char *optstring);
```



Počet CMD  
argumentov




Pole CMD  
argumentov

# Funkcia getopt


```
int getopt(int argc, char *argv[], const char *optstring);
```



Počet CMD  
argumentov



Pole CMD  
argumentov



Reťazec špecifikujúci  
platné prepínače

# Funkcia getopt

```
int getopt(int argc, char *argv[], const char *optstring);
```



Príklad reťazca *optstring*

```
char *optstring = ":ab:c:"
```

Reťazec špecifikujúci  
platné prepínače



# Funkcia getopt

```
char *optstring = ":ab:c:"
```

# Funkcia getopt

```
char *optstring = ":ab:c:"
```

Znak : na začiatku znamená, že  
getopt nevypíše chybové hlásenie

# Funkcia getopt

```
char *optstring = ":ab:c:"
```

Znak : na začiatku znamená, že  
getopt nevypíše chybové hlásenie

Prepínač *a*





# Funkcia getopt

```
char *optstring = ":ab:c:"
```

Znak `:` na začiatku znamená, že  
getopt nevypíše chybové hlásenie

Prepínač *a*

Prepínač *b* s  
povinným  
parametrom

# Funkcia getopt

```
char *optstring = ":ab:c:"
```

Znak `:` na začiatku znamená, že  
getopt nevypíše chybové hlásenie

Prepínač *a*

Prepínač *b* s  
povinným  
parametrom

Prepínač *c* s  
povinným  
parametrom

# Funkcia getopt

```
int getopt(int argc, char *argv[], const char *optstring);
```



Opakovaným volaním  
funkcia vracia písmená  
rozpoznaných prepínačov.

Ak funkcia rozpoznala všetky  
platné prepínače, vráti -1.

# Dokumentácia funkcie getopt

## getopt(3) — Linux manual page

[NAME](#) | [LIBRARY](#) | [SYNOPSIS](#) | [DESCRIPTION](#) | [RETURN VALUE](#) | [ENVIRONMENT](#) | [ATTRIBUTES](#) | [VERSIONS](#) | [STANDARDS](#) | [HISTORY](#) | [NOTES](#) | [EXAMPLES](#) | [SEE ALSO](#)

**getopt(3)** Library Functions Manual **getopt(3)**

### NAME [top](#)

getopt, getopt\_long, getopt\_long\_only, optarg, optind, opterr, optopt - Parse command-line options

### LIBRARY [top](#)

Standard C library (*libc*, *-lc*)

### SYNOPSIS [top](#)

```
#include <unistd.h>

int getopt(int argc, char *argv[],
           const char *optstring);

extern char *optarg;
extern int optind, opterr, optopt;
```

<https://man7.org/linux/man-pages/man3/getopt.3.html>



# Načítanie textu

- Po spustení programu a rozpoznaní CMD argumentov začne textový procesor načítavať text.

# Načítanie textu

- Po spustení programu a rozpoznaní CMD argumentov začne textový procesor načítavať text.
- Text sa bude načítavať po riadkoch zo štandardného vstupu (stdin).

# Načítanie textu

- Po spustení programu a rozpoznaní CMD argumentov začne textový procesor načítavať text.
- Text sa bude načítavať po riadkoch zo štandardného vstupu (stdin).
- Načítavanie skončí, keď používateľ zadá prázdny riadok.

# Načítanie textu

- Po spustení programu a rozpoznaní CMD argumentov začne textový procesor načítavať text.
- Text sa bude načítavať po riadkoch zo štandardného vstupu (stdin).
- Načítavanie skončí, keď používateľ zadá prázdny riadok.

```
#define EMPTY_LINE "\n"
```



# Načítanie textu

- V zadaní bude platiť, že maximálna dĺžka načítaného riadku je **1 000 znakov** (znaky riadku + znak nového riadku `'\n'`).
- Treba použiť pole s kapacitou 1001 znakov (kvôli ukončovaciemu znaku `'\0'`).

```
#define MAX_LINE 1000
```

# Funkcia fgets

- Na načítanie riadku odporúčame použiť knižničnú funkciu *fgets*.

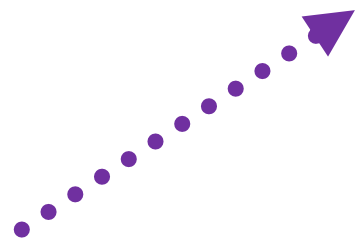
```
char * fgets ( char * str, int num, FILE * stream);
```

# Funkcia fgets

- Na načítanie riadku odporúčame použiť knižničnú funkciu *fgets*.

```
char * fgets ( char * str, int num, FILE * stream);
```

Načítaný reťazec  
(musí mať dostatočnú  
kapacitu).



# Funkcia fgets

- Na načítanie riadku odporúčame použiť knižničnú funkciu *fgets*.

```
char * fgets ( char * str, int num, FILE * stream);
```

Načítaný reťazec  
(musí mať dostatočnú  
kapacitu).

Do reťazca *str* sa zo streamu načíta  
*num-1* znakov alebo načítanie  
skončí po výskyte znaku '\n'.

# Funkcia fgets

- Na načítanie riadku odporúčame použiť knižničnú funkciu *fgets*.

```
char * fgets ( char * str, int num, FILE * stream );
```

Načítaný reťazec  
(musí mať dostatočnú  
kapacitu).

Do reťazca *str* sa zo streamu načíta  
*num-1* znakov alebo načítanie  
skončí po výskyte znaku '\n'.

Stream, v našom  
prípade *stdin*.

# Maximálna dĺžka slova

- V zadání budeme v texte hľadať slová.
- Maximálna dĺžka slova je 20 znakov (bez ukončovacieho znaku '\0').

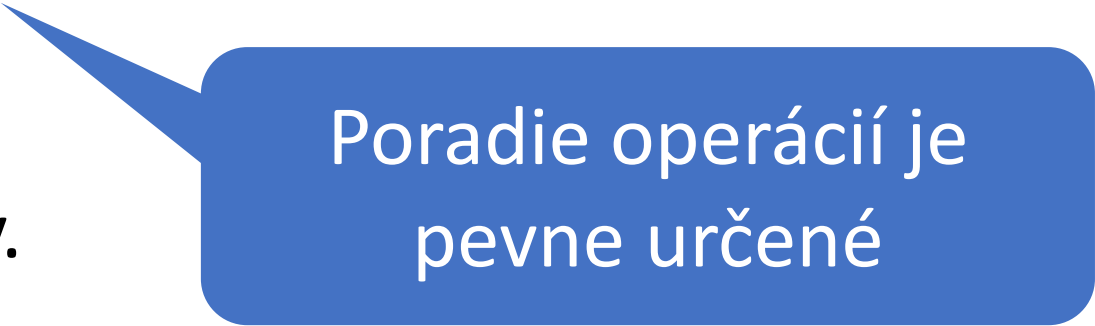
```
#define MAX_WORD 20
```

# Hlavné fázy programu

1. Spracovanie CMD argumentov a detekcia chybových situácií.
2. Načítanie textu.
3. Vstupné operácie.
4. Hľadanie a nahrádzanie slov.
5. Výstupné operácie.

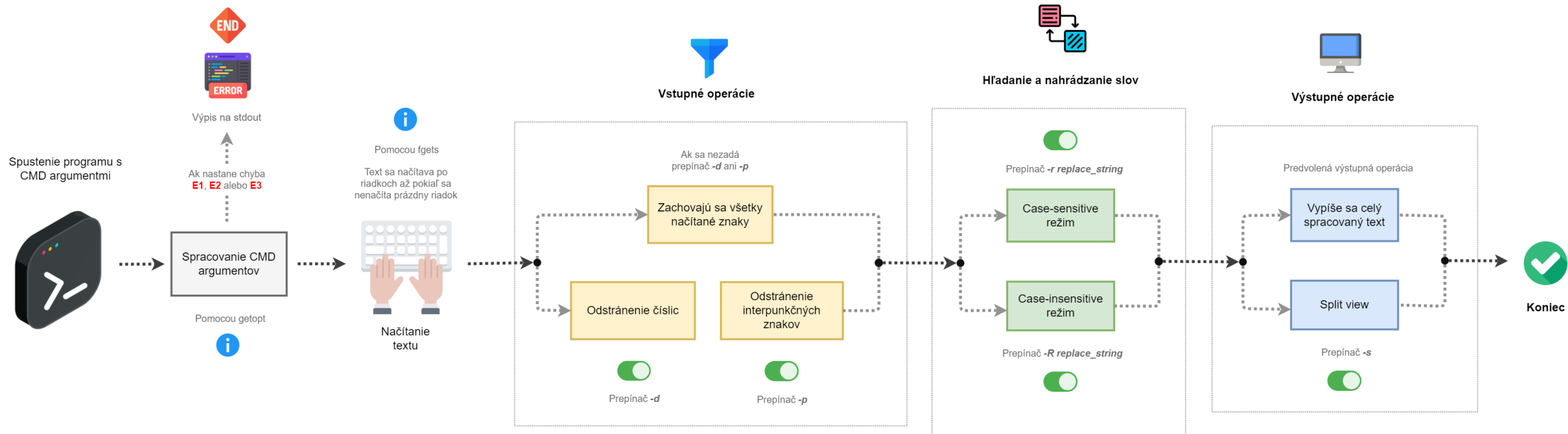
# Hlavné fázy programu

1. Spracovanie CMD argumentov a detekcia chybových situácií.
2. Načítanie textu.
3. Vstupné operácie.
4. Hľadanie a nahrádzanie slov.
5. Výstupné operácie.



Poradie operácií je  
pevne určené





Spustenie programu s  
CMD argumentmi



Spustenie programu s  
CMD argumentmi



Spracovanie CMD  
argumentov

Pomocou getopt



Spustenie programu s  
CMD argumentmi



Spracovanie CMD  
argumentov

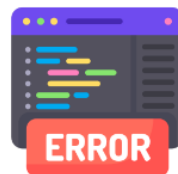
Pomocou getopt



Ak nastane chyba  
**E1**, **E2** alebo **E3**



Výpis na stdout



Spustenie programu s  
CMD argumentmi



Spracovanie CMD  
argumentov

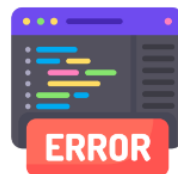
Pomocou getopt



Ak nastane chyba  
**E1**, **E2** alebo **E3**



Výpis na stdout



Pomocou fgets

Text sa načítava po  
riadkoch až pokiaľ sa  
nenačíta prázdny riadok



Načítanie  
textu





Pomocou fgets

Text sa načítava po  
riadkoch až pokiaľ sa  
nenačíta prázdny riadok



Načítanie  
textu



## Vstupné operácie



Pomocou fgets

Text sa načítava po  
riadkoch až pokiaľ sa  
nenačíta prázdny riadok



Načítanie  
textu





## Vstupné operácie



Pomocou fgets

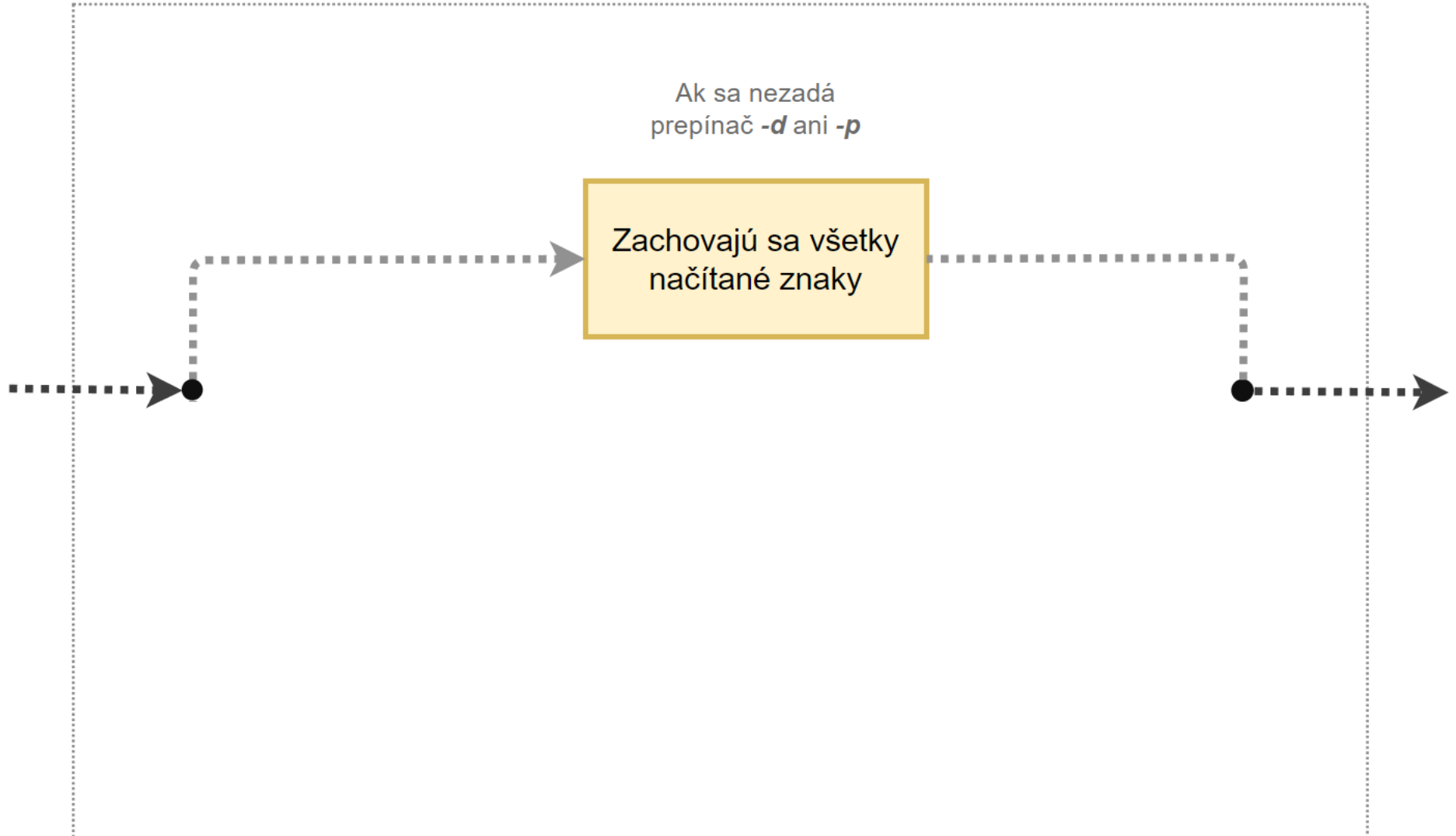
Text sa načítava po  
riadkoch až pokiaľ sa  
nenačíta prázdny riadok



Načítanie  
textu

Ak sa nezadá  
prepínač **-d** ani **-p**

Zachovajú sa všetky  
načítané znaky







## Vstupné operácie

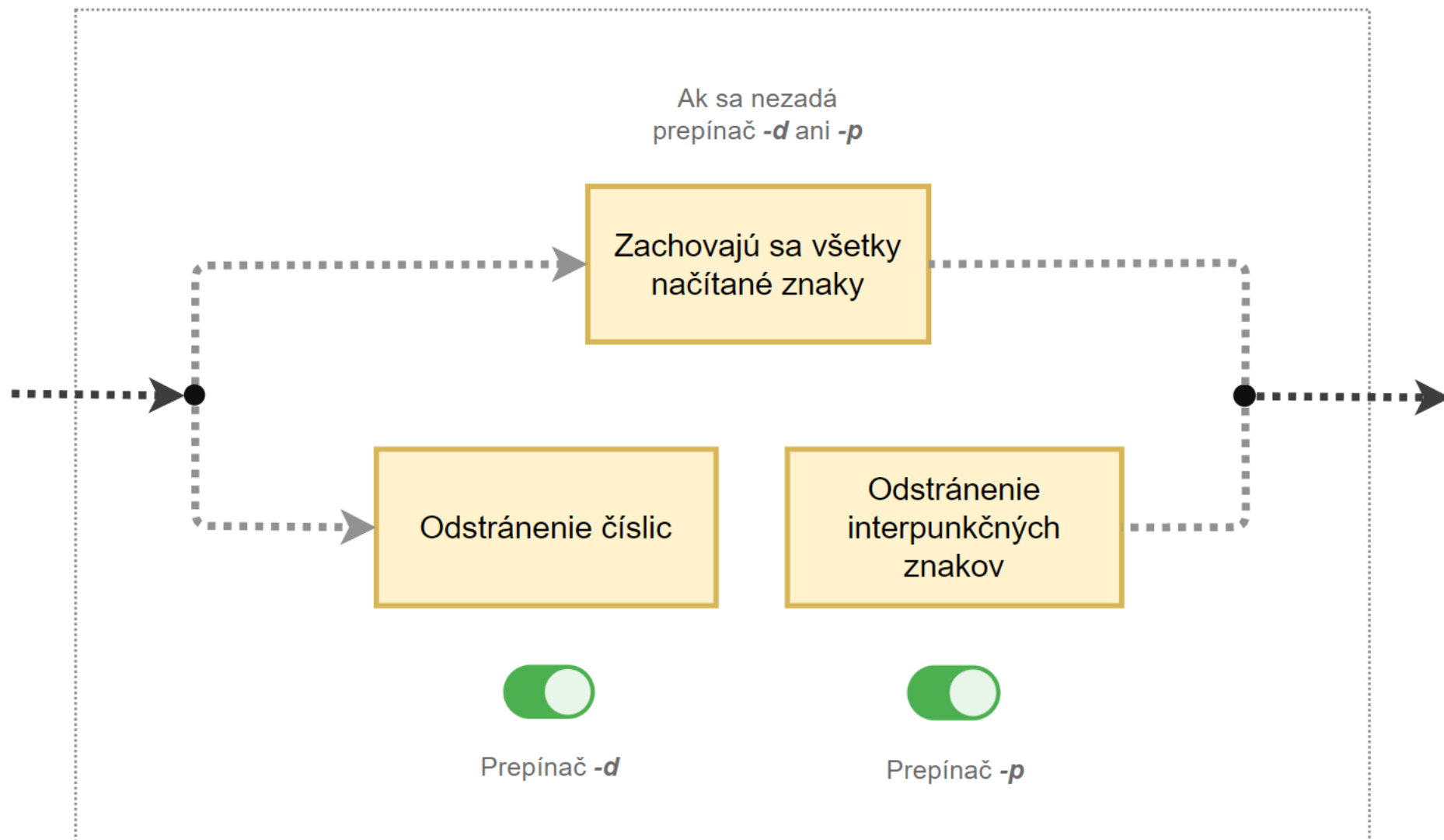


Pomocou fgets

Text sa načítava po  
riadkoch až pokiaľ sa  
nenačíta prázdny riadok



Načítanie  
textu





Hľadanie a nahrádzanie slov



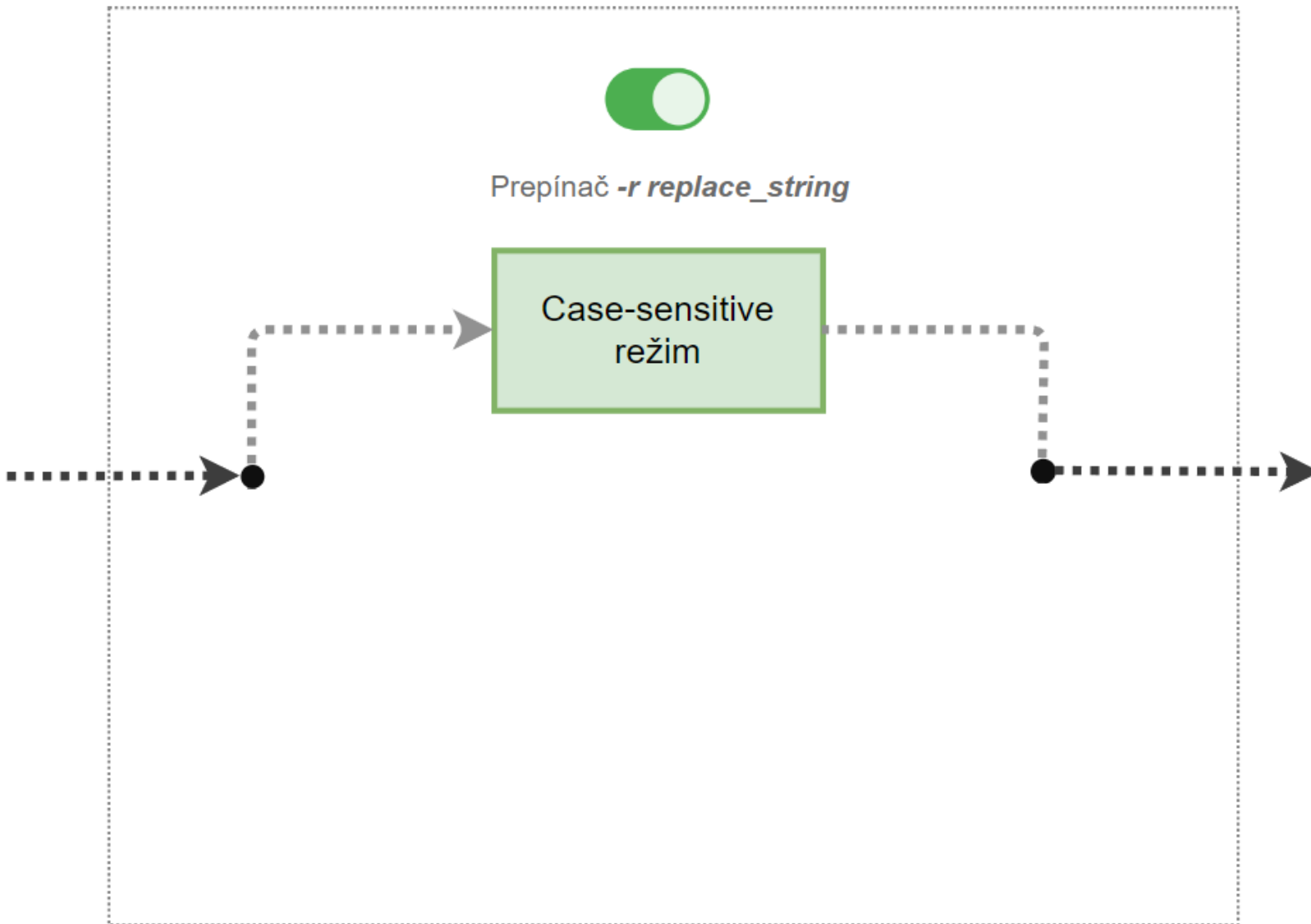


## Hľadanie a nahrádzanie slov



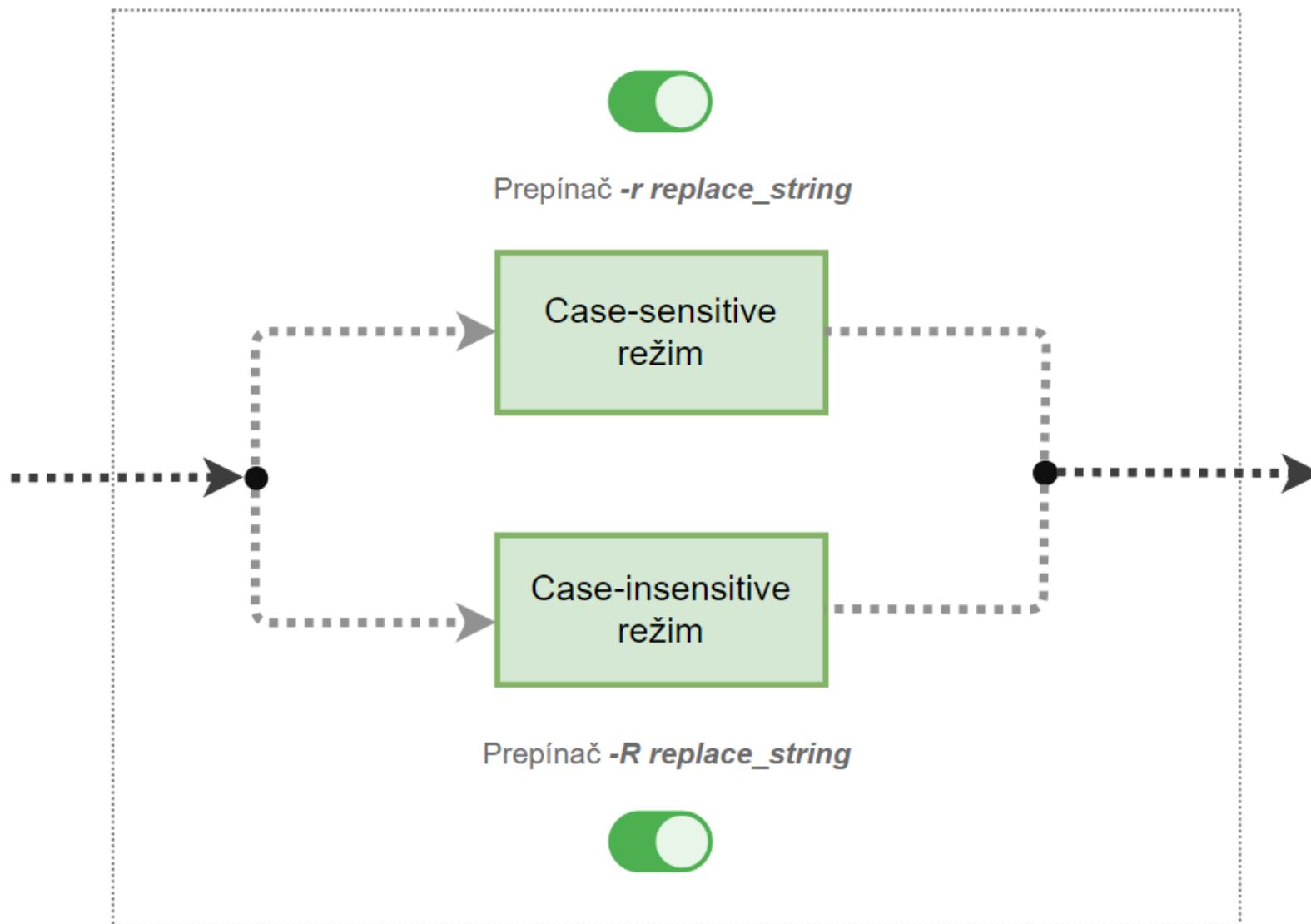
Prepínač *-r replace\_string*

Case-sensitive  
režim





## Hľadanie a nahrádzanie slov





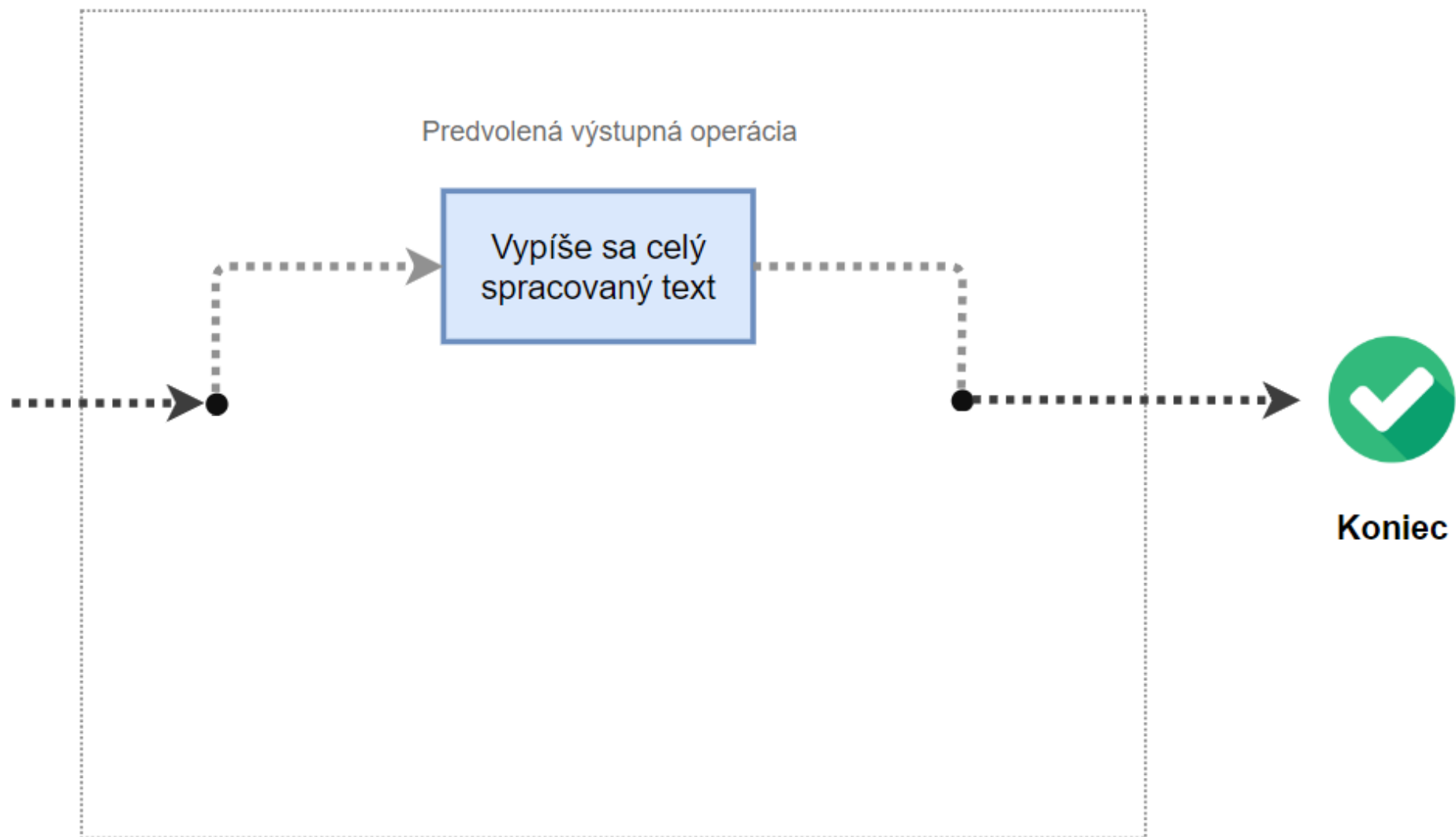
**Výstupné operácie**



**Koniec**

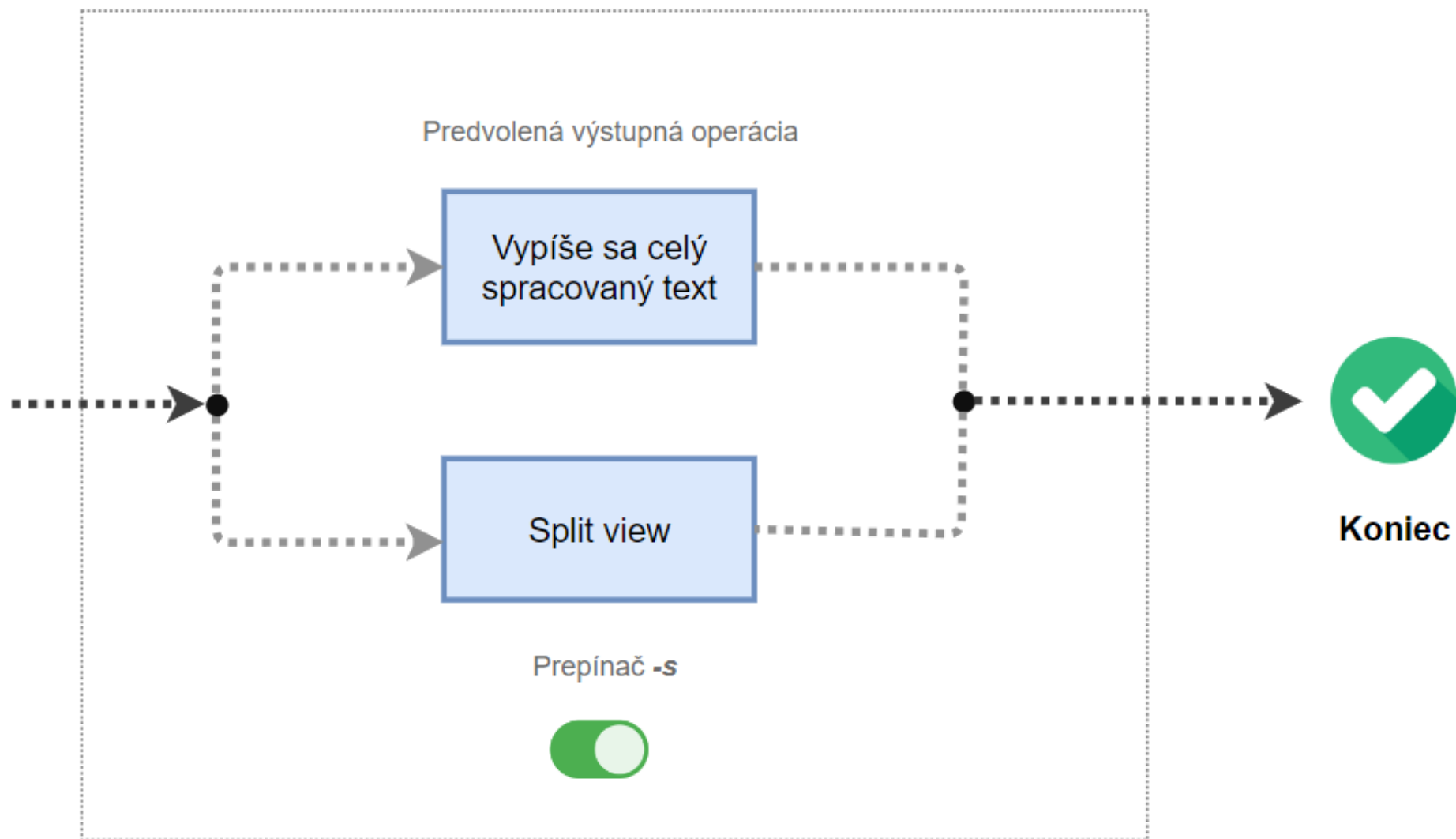


## Výstupné operácie





## Výstupné operácie



# Vstupné operácie

- Tieto operácie slúžia na zakázanie určitej skupiny znakov v načítanom texte.
- Rozlišujeme 3 typy vstupných operácií.
- Operácie dané prepínačom *-d* a *-p* sa môžu kombinovať.



# Vstupné operácie

## **Predvolená vstupná operácia:**

- Po načítaní textu sa v ňom ponechajú všetky znaky.
- Vykoná sa, keď nie je zadany prepínač *-d* ani *-p*.

z3.exe

*This is a sample text.*  
*123456*  
*.,-%\$§/+\*-*



This is a sample text.  
123456  
.,-%\$§/+\*-



# Vstupné operácie

## Prepínač *-d*:

- Pri zadaní tohto prepínača sa v načítanom texte odstráni číslce.
- Na overenie, či je znak číslcou môžeme použiť knižničnú funkciu *isdigit*.

z3.exe -d

*ahoj*  
*hello 123 world*  
*F1u2n3n4y5*  
*.3.3.3.*  
*a 999 b*



ahoj  
hello world  
Funny  
...  
a b



# Vstupné operácie

## Prepínač *-p*:

- Pri zadaní tohto prepínača sa v načítanom texte odstránia interpunkčné znaky.
- Na overenie, či je znak interpunkčný môžeme použiť knižničnú funkciu *ispunct*.

! " # \$ % & ' ( ) \* + , - . / : ; < = > ? @ [ \ ] ^ \_ ` { | } ~

z3.exe -p

*dogs & cats*  
*123456 123456*  
*what a world!*  
*<html>*  
*?a?1?5?W?*



dogs cats  
123456 123456  
what a world  
html  
a15W



z3.exe -d -p

z3.exe -p -d

*prett7y l8in45e123*

*.W.00.W.*

*?Si74m,p---le s2tup74id+\*\*.*



pretty line

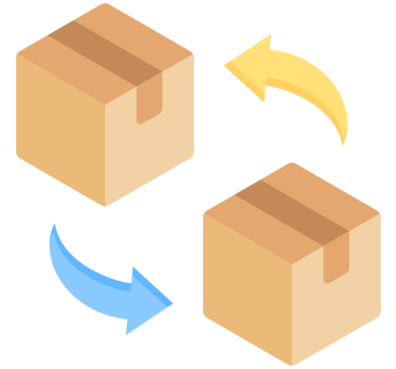
woow

Simple stupid



# Hľadanie a nahrádzanie slov

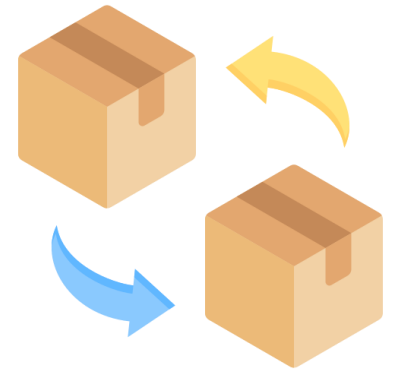
- Tieto operácie slúžia na nahradenie vyhovujúcich slov v texte iným zadaným slovom.





# Hľadanie a nahrádzanie slov

- Rozlišujeme 2 nahrádzacie režimy:
  - Case-sensitive (prepínač *-r*)
  - Case-insensitive (prepínač *-R*)
- Režimy sa **nesmú** kombinovať.



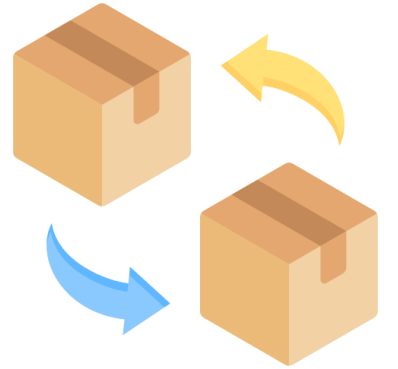
Tester ani nebude testovať prípady kombinácie oboch prepínačov súčasne.

# Hľadanie a nahrádzanie slov

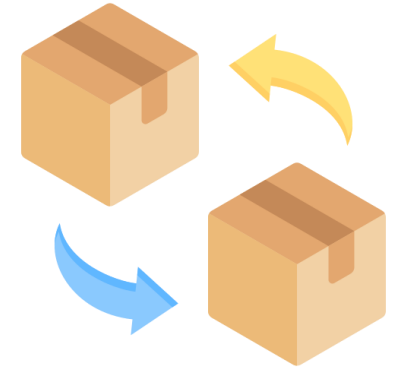
- Slová, ktoré vyhovujú kritériám hľadania sa nahradia pomocou reťazca, ktorý bol zadáný ako parameter prepínača *-r* alebo *-R*.

z3.exe *-r* replace\_string

z3.exe *-R* replace\_string



# Hľadanie a nahrádzanie slov



- Slová, ktoré vyhovujú kritériám hľadania sa nahradia pomocou reťazca, ktorý bol zadáný ako parameter prepínača *-r* alebo *-R*.

z3.exe *-r* replace\_string  
z3.exe *-R* replace\_string

Slová nahradzame  
reťazcom  
replace\_string

# Slovo

- Nahrádzanie prebieha po slovách.
- V zmysle tohto zadania budeme slovom označovať **ľubovoľnú alfabetickú postupnosť znakov**, ktorá je oddelená od okolitého textu ne-alfabetickými znakmi.



# Slovo

- Na zistenie, či je znak alfabetický môžete použiť knižničnú funkciu *isalpha*.
- Každý načítaný riadok textu vieme reprezentovať ako množinu slov  $W = \{w_1, w_2, w_3 \dots, w_K\}$ .



# Slovo



- Príklady riadkov a identifikovaných slov:

Milujem programovanie v jazyku C.

Milujem.programovanie.v.jazyku.C.

Milujem1programovanie2v3jazyku4C.



# Slovo



- Príklady riadkov a identifikovaných slov:

Milujem programovanie v jazyku C.

Milujem.programovanie.v.jazyku.C.

Milujem1programovanie2v3jazyku4C.

Milujem programovanie v jazyku C } Slová

# Slovo



- Príklady riadkov a identifikovaných slov:

Milujem programovanie v jazyku C.

Milujem.programovanie.v.jazyku.C.

Milujem1programovanie2v3jazyku4C.

Milujem programovanie v jazyku C } Slová



# Slovo



- Príklady riadkov a identifikovaných slov:

Milujem programovanie v jazyku C.

Milujem.programovanie.v.jazyku.C.

Milujem1programovanie2v3jazyku4C.

Milujem programovanie v jazyku C

} Slová

# Slovo



- Príklady riadkov a identifikovaných slov:

Zem@@sa[toci}78oko1o9\*+Slnka&.

Slová {

# Slovo



- Príklady riadkov a identifikovaných slov:

Zem@@sa[toci}78oko1o9\*+Slnka&.

Slová { Zem

# Slovo



- Príklady riadkov a identifikovaných slov:

Zem@@sa[toci}78oko1o9\*+Slnka&.

Slová { Zem sa

# Slovo



- Príklady riadkov a identifikovaných slov:

Zem@@sa[toci}78oko1o9\*+Slnka&.

Slová { Zem sa toci

# Slovo



- Príklady riadkov a identifikovaných slov:

Zem@@sa[toci}78okolo9\*+Slnka&.

Slová { Zem sa toci okolo

# Slovo



- Príklady riadkov a identifikovaných slov:

Zem@@sa[toci}78okolo9\*+Slnka&.

Slová { Zem sa toci okolo Slnka

# Slovo



- Príklady riadkov a identifikovaných slov:

z3.exe -d -p

==;D<23o9b?r+o z7v5<6.i%t;a~z|}i. \*n11a2\*d z9l(4om).

Slová {



# Slovo



- Príklady riadkov a identifikovaných slov:

z3.exe -d -p

==;D<23o9b?r+o z7v5<6.i%t;a~z|}i. \*n11a2\*d z9l(4om).

Slová {

# Slovo



- Príklady riadkov a identifikovaných slov:

z3.exe -d -p

==;D<23o9b?r+oz7v5<6.i%t;a~z|}i.\*n11a2\*d z9l(4om).

Slová { Dobro

# Slovo



- Príklady riadkov a identifikovaných slov:

z3.exe -d -p

==;D<23o9b?r+o z7v5<6.i%t;a~z|}i.\*n11a2\*d z9l(4om).

Slová { Dobro

# Slovo



- Príklady riadkov a identifikovaných slov:

z3.exe -d -p

==;D<23o9b?r+o z7v5<6.i%t;a~z|i.\*n11a2\*d z9l(4om).

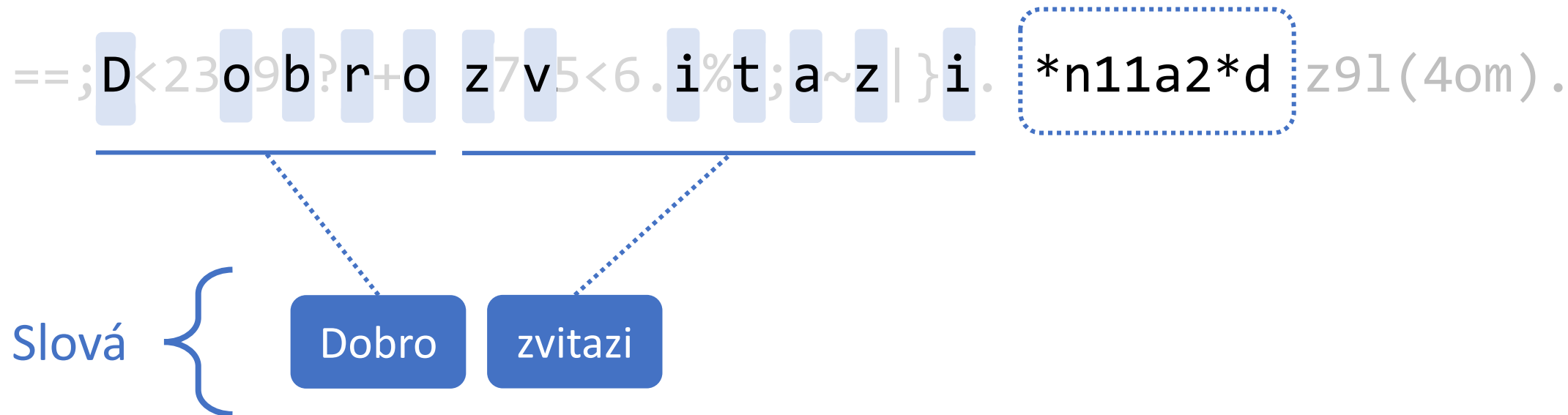
Slová { Dobro zvitazi

# Slovo



- Príklady riadkov a identifikovaných slov:

z3.exe -d -p

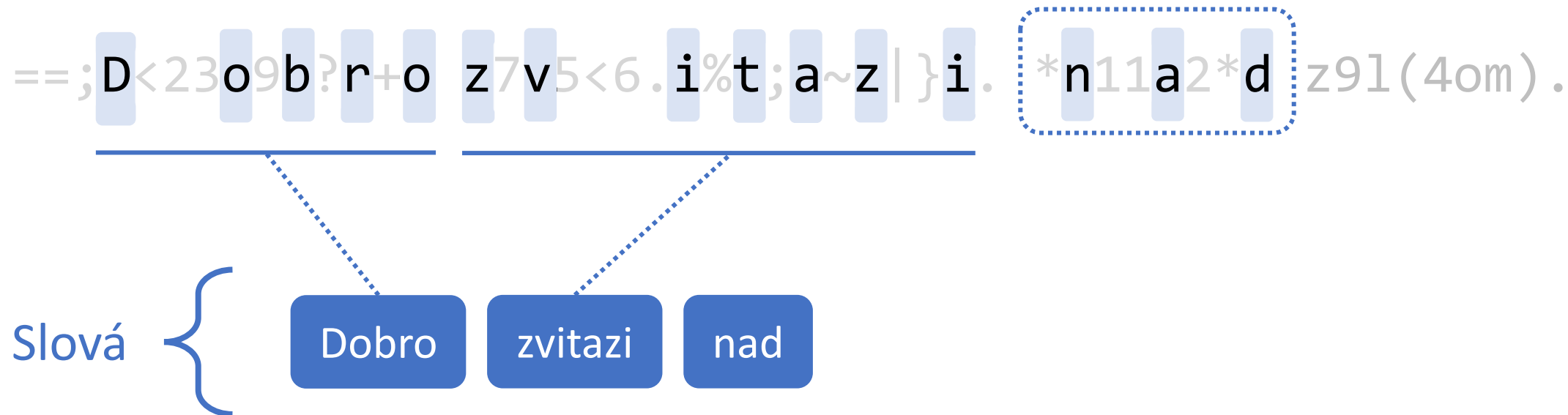


# Slovo



- Príklady riadkov a identifikovaných slov:

z3.exe -d -p

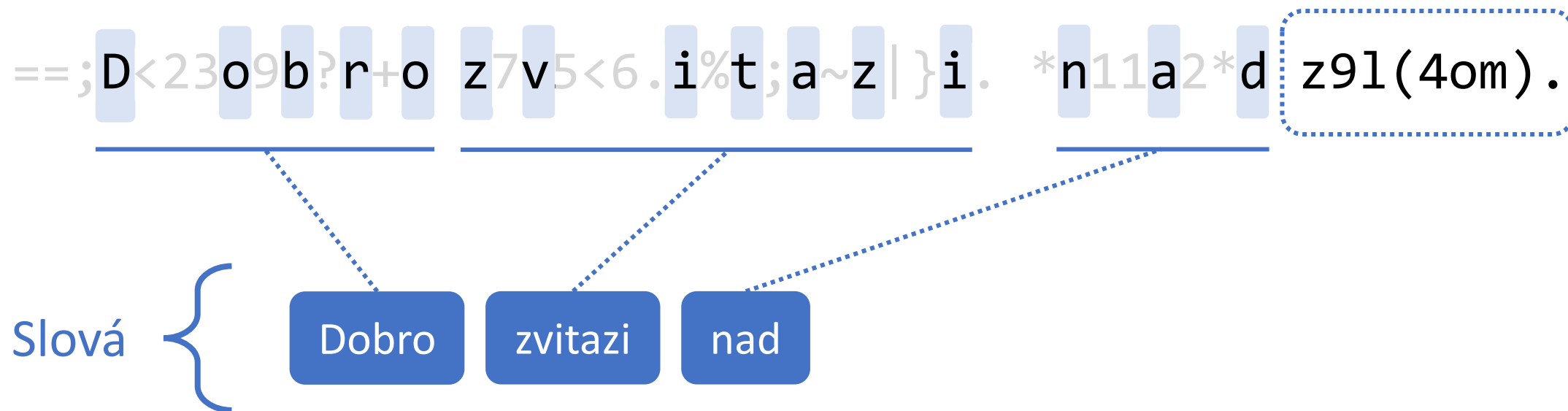


# Slovo



- Príklady riadkov a identifikovaných slov:

z3.exe -d -p

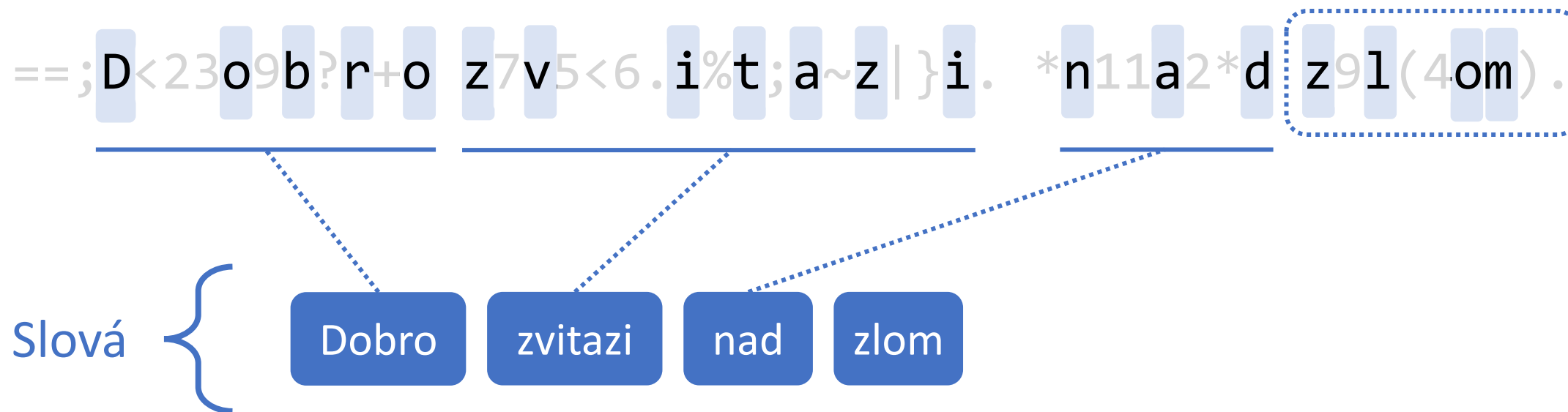


# Slovo



- Príklady riadkov a identifikovaných slov:

z3.exe -d -p



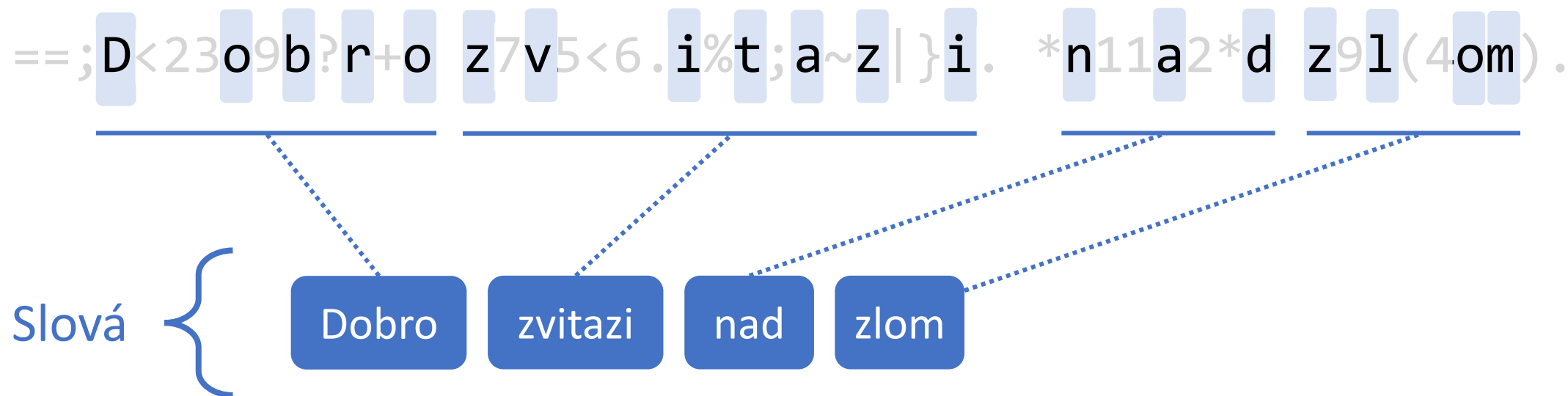


# Slovo



- Príklady riadkov a identifikovaných slov:

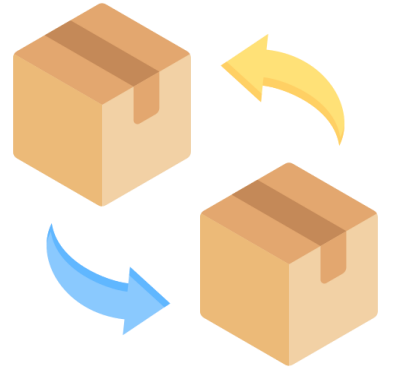
z3.exe -d -p



# Case-sensitive režim

- Formát spustenia režimu:

`z3.exe -r replace_string arg1 arg2 arg3`

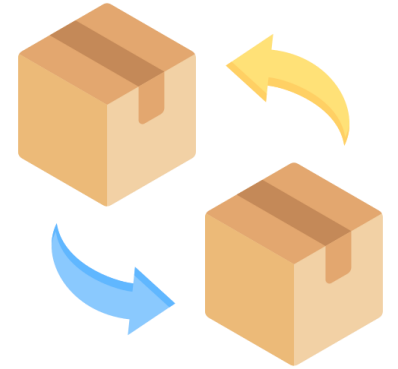


# Case-sensitive režim

- Formát spustenia režimu:

z3.exe -r replace\_string arg1 arg2 arg3

Reťazec, pomocou ktorého  
nahradzame slová.



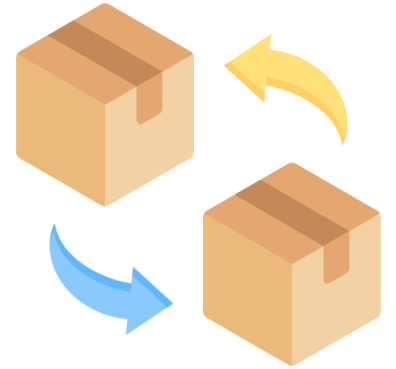
# Case-sensitive režim

- Formát spustenia režimu:

z3.exe -r replace\_string arg1 arg2 arg3

Reťazec, pomocou ktorého  
nahradzame slová.

Nepovinné non-option  
argumenty



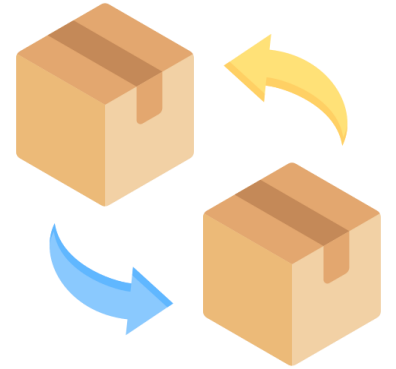
# Case-sensitive režim

Množinu non-option argumentov budeme označovať ako  $N = \{\text{arg1, arg2, arg3 ...}\}$

`z3.exe -r replace_string arg1 arg2 arg3`

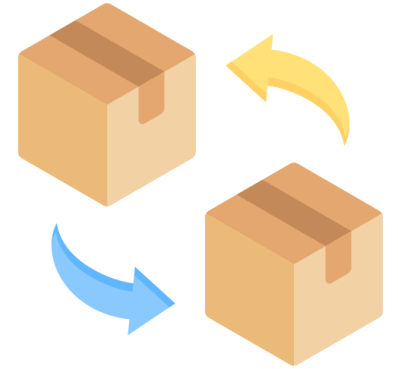
Reťazec, pomocou ktorého  
nahradzame slová.

Nepovinné non-option  
argumenty



# Case-sensitive režim

- Počas tejto operácie postupne prechádzame slová v množine  $W$  zľava doprava a nahrádzame ich pomocou reťazca *replace\_string*.



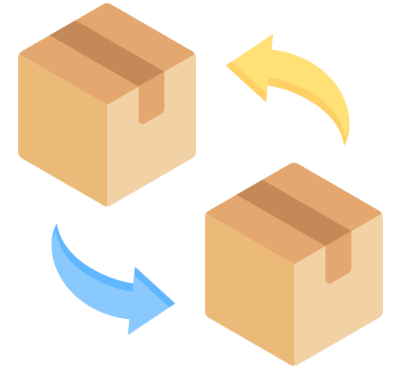
z3.exe -r replace\_string arg1 arg2 arg3



Množina  $N$

# Case-sensitive režim

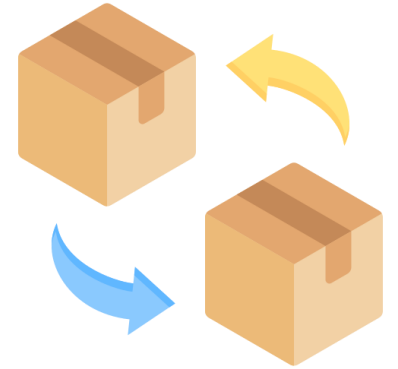
- Ostatné znaky riadku, ktoré netvoria slová zostávajú nezmenené.



z3.exe -r replace\_string arg1 arg2 arg3

Množina  $N$

# Case-sensitive režim



Množina  $N$

`z3.exe -r replace_string`  $\arg1$   $\arg2$   $\arg3$

Množina non-option argumentov  
predstavuje tzv. case-sensitive  
predpony.

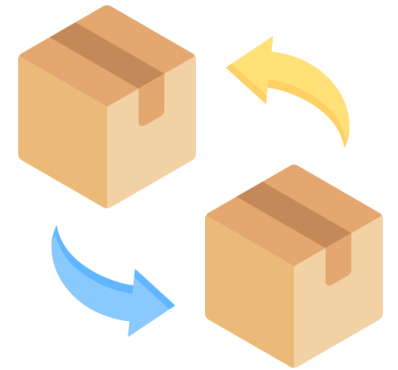


# Case-sensitive režim

Množina  $N$

`z3.exe -r replace_string`  $\arg_1$   $\arg_2$   $\arg_3$

Ktoré slová v načítanom riadku sú  
nahradené pomocou reťazca  
*replace\_string*?



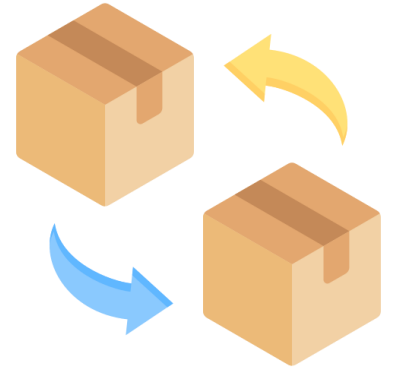
# Case-sensitive režim

Množina  $N$

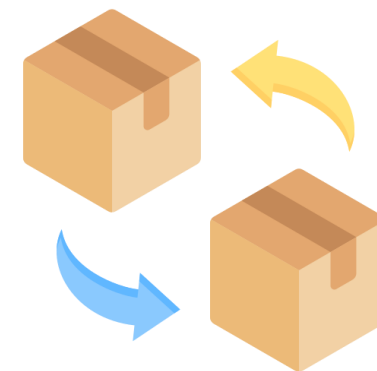
`z3.exe -r replace_string`  $\arg1$   $\arg2$   $\arg3$

Zadaný text:

*Abc, SLOVO\_123Abcedg, dnes? ... \*\*AbcDEFGH\*\**



# Case-sensitive režim



Množina  $N$

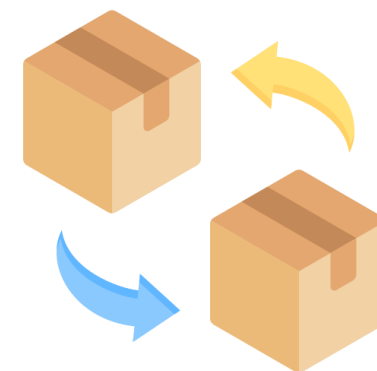
`z3.exe -r replace_string`  $\arg1$   $\arg2$   $\arg3$

Zadaný text:

$w_1$   $w_2$   $w_3$   $w_4$   $w_5$

*Abc*, *SLOVO*\_123*Abcedg*, *dnes?* ... \*\**AbcDEFGH*\*\*

# Case-sensitive režim



Množina  $N$

`z3.exe -r replace_string`  $\arg1$   $\arg2$   $\arg3$

Zadaný text:

*Abc*, *SLOVO*\_123*Abcedg*, *dnes?* ... *\*\*AbcDEFGH\*\**

$w_1$

$w_2$

$w_3$

$w_4$

$w_5$

Množina  $W$

# Case-sensitive režim



Ak je množina  $N$  neprázdná:

# Case-sensitive režim



Ak je množina  $N$  neprázdná:

Pre každé jedno slovo  $w_i$  v množine  $W$  hľadáme jeho case-sensitive predponu v množine  $N$ .

# Case-sensitive režim



Ak je množina  $N$  neprázdna:

Pre každé jedno slovo  $w_i$  v množine  $W$  hľadáme jeho case-sensitive predponu v množine  $N$ .

Ak pre skúmané slovo  $w_i$  nájdeme case-sensitive predponu v množine  $N$ , potom dôjde k nahradeniu slova  $w_i$  pomocou reťazca *replace\_string*.

# Case-sensitive režim



Ak je množina  $N$  prázdna:



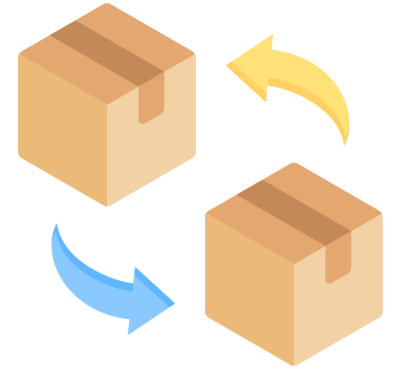
# Case-sensitive režim

V tomto prípade nedochádza k hľadaniu predpony. Namiesto toho sa nahradí každé jedno slovo z množiny  $W$  pomocou *replace\_string*.



Ak je množina  $N$  prázdna:

# Case-sensitive režim



z3.exe -r Vianoce abc ABC Abc

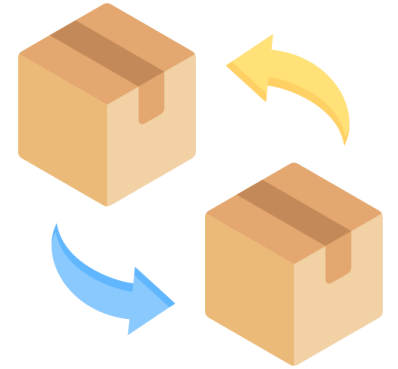
Zadaný text:

Množina  $N$

Abc, SLOVO\_123Abcedg, dnes? ... \*\*AbcDEFGH\*\*

$w_1$

# Case-sensitive režim



z3.exe -r Vianoce abc ABC Abc



Zadaný text:

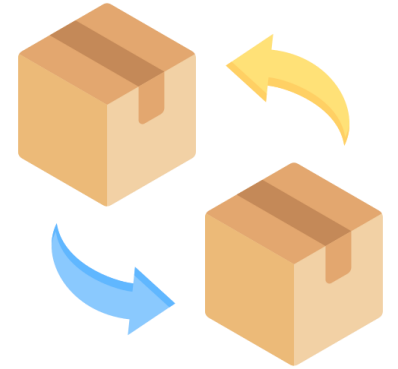
Množina  $N$

Abc, SLOVO\_123Abcedg, dnes? ... \*\*AbcDEFGH\*\*

$w_1$

Hľadáme case-sensitive  
predponu slova  $w_1$  v množine  $N$ .

# Case-sensitive režim



z3.exe -r Vianoce abc ABC Abc



Množina  $N$

Zadaný text:

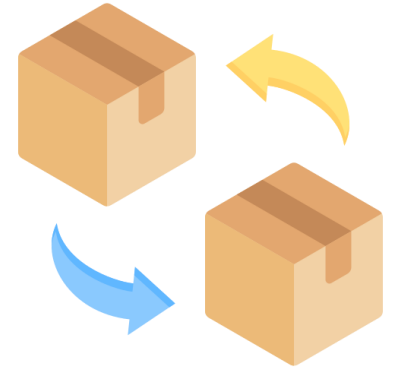
*Abc, SLOVO\_123Abcedg, dnes? .*

$w_1$

Řetazec "Abc" je case-sensitive  
predponou slova  $w_1$ .



# Case-sensitive režim



z3.exe -r Vianoce abc ABC Abc

$n_3$

└──────────┘

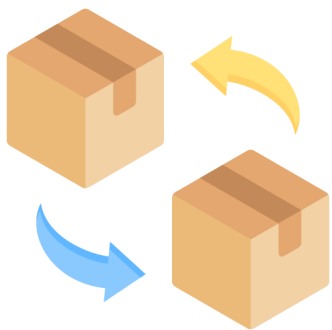
Zadaný text:

Množina  $N$

Abc, SLOVO\_123Abcedg, dnes? ... \*\*AbcDEFGH\*\*

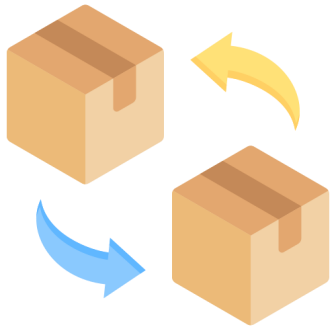
$w_1$

# Case-sensitive režim



Ako prebieha  
nahrádzanie slova?

# Case-sensitive režim

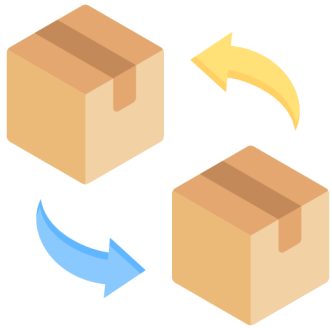


Ako prebieha  
nahrádzanie slova?

1

Postupne nahrádzame znaky  
skúmaného slova  $w_i$  znakmi  
reťazca *replace\_string* zľava doprava.

# Case-sensitive režim



Ako prebieha  
nahrádzanie slova?

1

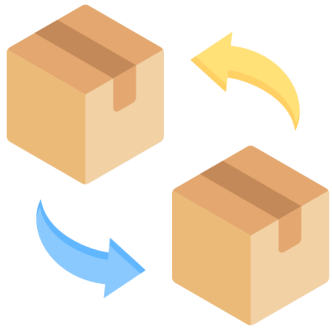
Postupne nahrádzame znaky  
skúmaného slova  $w_i$  znakmi  
reťazca *replace\_string* zľava doprava.

2

Ak je *replace\_string* kratší ako slovo  
 $w_i$ , nahradíme len príslušnú časť  
slova  $w_i$  a jeho zvyšok ponecháme  
nezmenený.



# Case-sensitive režim



Ako prebieha  
nahrádzanie slova?

1

Postupne nahrádzame znaky skúmaného slova  $w_i$  znakmi reťazca *replace\_string* zľava doprava.

2

Ak je *replace\_string* kratší ako slovo  $w_i$ , nahradíme len príslušnú časť slova  $w_i$  a jeho zvyšok ponecháme nezmenený.

3

Ak je *replace\_string* dlhší ako slovo  $w_i$ , nahrádzanie prebieha len po koniec slova  $w_i$ .

# Case-sensitive režim

EXAMPLE

z3.exe -r Vianoce abc ABC Abc



Zadaný text:

Množina  $N$

Abc, SLOVO\_123Abcedg, dnes? ... \*\*AbcDEFGH\*\*

$w_1$

# Case-sensitive režim

EXAMPLE

z3.exe -r Vianoce abc ABC Abc



Množina  $N$

Zadaný text:

✓  
Via, SLOVO\_123Abcedg, dnes? ... \*\*AbcDEFGH\*\*

$w_1$

Slovo  $w_1$ ="Abc" sme nahradili slovom "Vianoce".  
Max. do dĺžky 3.

# Case-sensitive režim

EXAMPLE

z3.exe -r Vianoce abc ABC Abc



Množina  $N$

Zadaný text:

✓  
*Via*, *SL0V0*\_123Abcedg, dnes? ... \*\*AbcDEFGH\*\*

$w_1$

$w_2$

# Case-sensitive režim

EXAMPLE

z3.exe -r Vianoce abc ABC Abc



Množina  $N$

Zadaný text:

✓  
Via, SLOVO\_123Abcedg, dnes? ... \*\*AbcDEFGH\*\*

$w_1$

$w_2$

Slovo  $w_2$ ="SLOVO" nemá v množine  $N$  case-sensitive predponu.  
Náhrada sa nevykoná.

# Case-sensitive režim



z3.exe -r Vianoce abc ABC Abc



Množina  $N$

Zadaný text:

✓  
*Via*, SLOVO\_123*Abcedg*, dnes? ... **\*\*AbcDEFGH\*\***

$w_1$

$w_3$

# Case-sensitive režim

EXAMPLE

z3.exe -r Vianoce abc ABC Abc



Množina  $N$

Zadaný text:

✓  
Via, SLOVO\_123 ✓  
Vianoc, dnes? ... \*\*AbcDEFGH\*\*

$w_1$

$w_3$

Slovo  $w_3$ ="Abcedg" sme nahradili slovom "Vianoce".  
Max. do dĺžky 6.

# Case-sensitive režim

EXAMPLE

z3.exe -r Vianoce abc ABC Abc



Množina  $N$

Zadaný text:

✓ *Via*, SLOVO\_123 ✓ *Vianoc*, *dnes?* ... \*\*AbcDEFGH\*\*

$w_1$

$w_3$

$w_4$



# Case-sensitive režim

EXAMPLE

z3.exe -r Vianoce abc ABC Abc



Množina  $N$

Zadaný text:

✓ *Via*, SLOVO\_123 ✓ *Vianoc*, *dnes?* ... \*\*AbcDEFGH\*\*

$w_1$

$w_3$

$w_4$

Slovo  $w_4$ ="dnes" nemá v množine  $N$  case-sensitive predponu.  
Náhrada sa nevykoná.

# Case-sensitive režim

EXAMPLE

z3.exe -r Vianoce abc ABC Abc



Množina  $N$

Zadaný text:

✓ *Via*, SLOVO\_123 ✓ *Vianoc*, dnes? ... \*\**AbcDEFGH*\*\*

$w_1$

$w_3$

$w_5$

# Case-sensitive režim

EXAMPLE

z3.exe -r Vianoce abc ABC Abc



Zadaný text:

Množina  $N$

✓ Via, SLOVO\_123 ✓ Vianoc, dnes? ... \*\* VianoceH \*

$w_1$

$w_5$

Slovo  $w_5$ ="AbcDEFGH" sme nahradili slovom "Vianoce".  
Max. do dĺžky 7.

# Case-sensitive režim

EXAMPLE

z3.exe -r Vianoce abc ABC Abc



Množina  $N$

Zadaný text:

✓  
*Via*, SLOVO\_123 ✓  
*Vianoc*, dnes? ... \*\* ✓  
*VianoceH*\*

$w_1$

$w_3$

$w_5$

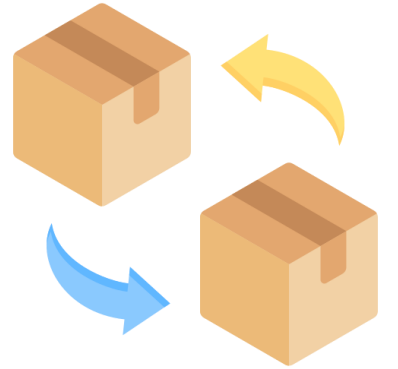
# Case-insensitive režim

- Formát spustenia režimu:

z3.exe -R replace\_string arg1 arg2 arg3

Reťazec, pomocou ktorého  
nahradzame slová.

Nepovinné non-option  
argumenty



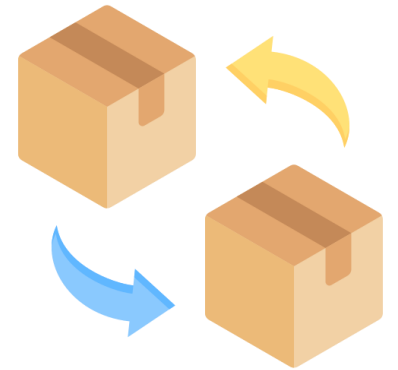
# Case-insensitive režim

Množinu non-option argumentov budeme označovať ako  $N = \{\text{arg1}, \text{arg2}, \text{arg3} \dots\}$

z3.exe -R replace\_string arg1 arg2 arg3

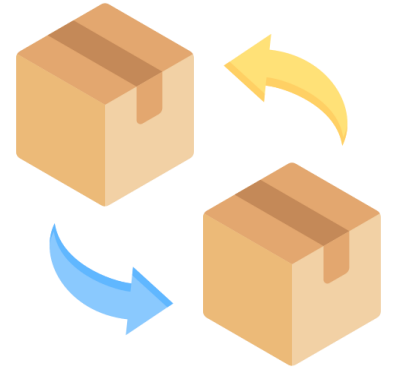
Reťazec, pomocou ktorého nahradzame slová.

Nepovinné non-option argumenty



# Case-insensitive režim

- Tento režim je rovnaký ako case-sensitive režim.
- Jediný rozdiel je v tom, že množina  $N$  obsahuje **case-insensitive** prepdkpony.



z3.exe -R replace\_string arg1 arg2 arg3



Množina N

# Case-insensitive režim



z3.exe -R "Hello world" Pr

Množina  $N$

Zadaný text:

*pRosim.p?PROGRAM 9pragmaticky @@@ 123-please*



# Case-insensitive režim

EXAMPLE

z3.exe -R "Hello world" Pr

Množina  $N$

Zadaný text:

*pRosim.p?PROGRAM 9pragmaticky @@@ 123-please*

$w_1$

# Case-insensitive režim

EXAMPLE

z3.exe -R "Hello world" Pr

Množina  $N$

Zadaný text:

✓  
Hello.p?PROGRAM 9pragmaticky @@@ 123-please

$w_1$

Slovo  $w_1$ ="pRosim" sme nahradili slovom "Hello world".  
Max. do dĺžky 6.

# Case-insensitive režim

EXAMPLE

z3.exe -R "Hello world" Pr

Množina  $N$

Zadaný text:

✓  
*Hello* *.p?PROGRAM 9pragmaticky @@@ 123-please*

$w_1$   $w_2$

# Case-insensitive režim



z3.exe -R "Hello world" Pr

Množina  $N$

Zadaný text:

✓  
*Hello* .p?PROGRAM 9pragmaticky @@@ 123-please

$w_1$   $w_3$

# Case-insensitive režim

EXAMPLE

z3.exe -R "Hello world" Pr

Množina  $N$

Zadaný text:

✓ Hello .p? ✓ Hello w 9pragmaticky @@@ 123-please

$w_1$

$w_3$

Slovo  $w_3$ ="PROGRAM" sme nahradili slovom "Hello world".  
Max. do dĺžky 7.

# Case-insensitive režim

EXAMPLE

z3.exe -R "Hello world" Pr



Množina  $N$

Zadaný text:

✓  
*Hello* ✓  
*.p?Hello w* *9pragmaticky* @@@ 123-please  
 $w_1$   $w_3$   $w_4$

# Case-insensitive režim

EXAMPLE

z3.exe -R "Hello world" Pr

Množina  $N$

Zadaný text:

✓ Hello .p? ✓ Hello w 9 ✓ Hello world @@@ 123-please

$w_1$   $w_3$   $w_4$

Slovo  $w_4$ ="pragmaticky" sme nahradili slovom "Hello world".  
Max. do dĺžky 11.

# Case-insensitive režim

EXAMPLE

z3.exe -R "Hello world" Pr



Množina  $N$

Zadaný text:

✓  
Hello .p? ✓ Hello w 9 ✓ Hello world @@@ 123-please

$w_1$

$w_3$

$w_4$

$w_5$



# Case-insensitive režim

EXAMPLE

z3.exe -R "Hello world" Pr



Množina  $N$

Zadaný text:

✓  
Hello .p? ✓ Hello w 9 ✓ Hello world @@@ 123-please  
 $w_1$   $w_3$   $w_4$

z3.exe -R "Hello world" Pr

EXAMPLE

*pRosim.p?PROGRAM 9pragmaticky @@@ 123-please*

    
*Hello*.p?*Hello w*9*Hello world* @@@ 123-please  
 $w_1$   $w_3$   $w_4$

# Výstupné operácie

- Tieto operácie slúžia na výpis výsledku spracovania textu.

# Výstupné operácie

Rozlišujeme 2 výstupné operácie:

- Predvolená výstupná operácia (ak nie je zadany prepínač **-s**)
- Prepínač **-s** (tzv. split view)

# Predvolená výstupná operácia

- Jej úlohou je vypísať celý text po spracovaní (t.j. po aplikovaní vstupných operácií a nahradení slov).

# Predvolená výstupná operácia



```
z3.exe -r "Lucky Number" sm The GOOD
```

```
Smile:)smile  
the98end  
GOOD Morning  
GOODness  
456TheDestroyer TheDark-Lord
```

Vstupný text

```
Smile:)Lucky  
the98end  
Luck Morning  
Lucky Nu  
456Lucky Number Lucky N-Lord
```

Výstup (*stdout*)

# Predvolená výstupná operácia

EXAMPLE

```
z3.exe -d -p
```

12345A6789

. - , \* & ^ % \$ # f @ !

Programovanie

1D\*o-vo-l0en156k,a+

Vstupný text

A

f

Programovanie

Dovolenka

Výstup (*stdout*)

# Predvolená výstupná operácia



```
z3.exe -p -R "Bruce Willis" sUpeR
```

```
superBowL  
5SU,PER,ma%%n9 sUpe  
slovak republic  
sUpernatural  
Supe  
sUpe9r  
supper lunch ssuper ..s?u?PE?r
```

Vstupný text

```
Bruce Wil  
5Bruce Wi9 sUpe  
slovak republic  
Bruce Willis  
Supe  
sUpe9r  
supper lunch ssuper Bruce
```

Výstup (*stdout*)



# Split view

- Táto operácia sa vykoná, ak je zadany prepínač **-s**.
- Výstup v tomto prípade bude rozdelený na dve časti:
  - **Ľavá časť** – slová textu pred vykonaním nahradenia slov (po aplikácii vstupných operácií !!!)
  - **Pravá časť** – reťazce, ktoré zo slov vznikli po nahradení.

# Split view

- Táto operácia sa vykoná, ak je zadany prepínač `-s`.
- Výstup v tomto prípade bude rozdelený na dve časti:
  - **Ľavá časť** – **slová** textu pred vykonaním nahradenia slov (po aplikácii vstupných operácií)
  - **Pravá časť** – reťazce nahradení.

Nezabudnite: slovo je v zmysle zadania alfabetická postupnosť znakov oddelená od okolitého textu ne-alfabetickými znakmi.

# Split view

EXAMPLE

```
z3.exe -s
```

Vstupný text

Výstup (*stdout*)

# Split view

EXAMPLE

```
z3.exe -s
```

```
nikto596nie.je Dokonały  
123  
__?__  
a b c d
```

Vstupný text

```
1. nikto      :nikto  
1. nie        :nie  
1. je         :je  
1. Dokonały:Dokonały  
4. a          :a  
4. b          :b  
4. c          :c  
4. d          :d
```

Výstup (*stdout*)

1. nikto :nikto

1. nie :nie

1. je :je

1. Dokonały:Dokonały

4. a :a

4. b :b

4. c :c

4. d :d

Poradové číslo  
riadku vo  
vstupnom texte.

1.	nikto	:nikto
1.	nie	:nie
1.	je	:je
1.	Dokonaľy	:Dokonaľy
4.	a	:a
4.	b	:b
4.	c	:c
4.	d	:d

1.	nikto	:nikto
1.	nie	:nie
1.	je	:je
1.	Dokonały	:Dokonały
4.	a	:a
4.	b	:b
4.	c	:c
4.	d	:d

1 medzera.

1. nikto :nikto  
1. nie :nie  
1. je :je  
1. Dokonaľy:Dokonaľy  
4. a :a  
4. b :b  
4. c :c  
4. d :d

Slová vo vstupnom  
texte pred vykonaním  
nahrádzania (po  
vykonaní vstupných  
operácií !!!)



1. nikto	:nikto
1. nie	:nie
1. je	:je
1. Dokonały	:Dokonały
4. a	:a
4. b	:b
4. c	:c
4. d	:d

Oddeľovací znak :

1. nikto	:nikto
1. nie	:nie
1. je	:je
1. Dokonały	:Dokonały
4. a	:a
4. b	:b
4. c	:c
4. d	:d

Reťazce, ktoré zo  
slov vznikli po  
nahradení.

Zarovnaný doľava

Šírka stĺpca = dĺžka  
najdlhšieho reťazca  
v stĺpci.

1.	nikto	:nikto
1.	nie	:nie
1.	je	:je
1.	Dokonaľy	:Dokonaľy
4.	a	:a
4.	b	:b
4.	c	:c
4.	d	:d

Šírka stĺpca

1.	nikto	:nikto
1.	nie	:nie
1.	je	:je
1.	Dokonalý	:Dokonalý
4.	a	:a
4.	b	:b
4.	c	:c
4.	d	:d

Zarovnaný doľava



Šírka stĺpca

Šírka stĺpca = dĺžka  
najdlhšieho reťazca  
v stĺpci.

# Split view

```
z3.exe -d -s
```

```
7777caso-priestor  
fot145obu87nk11a@fotopasca  
951 123  
X Y .Z.  
++  
Miestnost DE-150
```

Vstupný text

EXAMPLE

```
1. caso      :caso  
1. priestor  :priestor  
2. fotobunka:fotobunka  
2. fotopasca:fotopasca  
4. X         :X  
4. Y         :Y  
4. Z         :Z  
6. Miestnost:Miestnost  
6. DE        :DE
```

Výstup (stdout)

# Split view

EXAMPLE

```
z3.exe -p -s -r "Noc je dlha" den sve
```

```
Den-svetov?eho mieru.  
Svetovy den mieru.  
456 95123  
svetadiel.  
(d)(e)(n)ominator5Denis
```

Vstupný text

```
1. Densvetoveho:Densvetoveho  
1. mieru      :mieru  
2. Svetovy    :Svetovy  
2. den        :Noc  
2. mieru      :mieru  
4. svetadiel  :Noc je dl  
5. denominator :Noc je dlha  
5. Denis      :Denis
```

Výstup (stdout)

# Split view

EXAMPLE

```
z3.exe -d -p -s -R AAAAA nova
```

```
N[0]V5A du1bni[c]3a33  
$n$$o$V7$ no$5vink32$a  
764*+7&  
n0vANadej  
N-;0;VA; {no} 156 ??? NO#vANoV#A#
```

Vstupný text

```
1. NOVA      :AAAA  
1. dubnica   :dubnica  
2. noV       :noV  
2. novinka   :novinka  
4. n0vANadej:AAAAAadej  
5. NOVA      :AAAA  
5. no        :no  
5. NOvANoVA  :AAAAAoVA
```

Výstup (*stdout*)

# Chybové situácie

Budeme rozlišovať tieto chybové situácie:

- E1 - neplatný prepínač





# Chybové situácie



Budeme rozlišovať tieto chybové situácie:

- E1 - neplatný prepínač

Používateľ spustí program a zadá prepínač, ktorý nie je platný.

Písmená platných prepínačov: d, p, r, R, s

# Chybové situácie



Budeme rozlišovať tieto chybové situácie:

- E1 - neplatný prepínač

Program vypíše na stdout hlásenie „E1“ a korektne skončí.

# Chybové situácie

Budeme rozlišovať tieto chybové situácie:

- E1 - neplatný prepínač

```
z3.exe -d -e -r ahoj predpona
```



# Chybové situácie



Budeme rozlišovať tieto chybové situácie:

- E1 - neplatný prepínač

```
z3.exe -d -e -r ahoj predpona
```



Neznámy  
prepínač -e.

# Chybové situácie

Budeme rozlišovať tieto chybové situácie:

- E1 - neplatný prepínač
- E2 - chýbajúci povinný parameter prepínača



# Chybové situácie



Budeme rozlišovať tieto chybové situácie:

- E1 - neplatný prepínač
- E2 - chýbajúci povinný parameter prepínača

Používateľ spustí program s prepínačom `-r` alebo `-R`, ale nezadá jeho povinný parameter

# Chybové situácie



Budeme rozlišovať tieto chybové situácie:

- E1 - neplatný prepínač
- E2 - chýbajúci povinný parameter prepínača

Program vypíše na stdout hlásenie „E2“ a korektne skončí.

# Chybové situácie

Budeme rozlišovať tieto chybové situácie:

- E1 - neplatný prepínač
- E2 - chýbajúci povinný parameter prepínača



Chýba parameter  
prepínača -r

z3.exe -p -d -s -r





# Chybové situácie



Budeme rozlišovať tieto chybové situácie:

- E1 - neplatný prepínač
- E2 - chýbajúci povinný parameter prepínača
- E3 - parameter prepínača  $-r$  alebo  $-R$  nemá platnú dĺžku

# Chybové situácie



Budeme rozlišovať tieto chybové situácie:

- E1 - neplatný prepínač
- E2 - chýbajúci povinný parameter prepínača
- E3 - parameter prepínača  $-r$  alebo  $-R$  nemá platnú dĺžku

Platná dĺžka je v intervale  $\langle 1,20 \rangle$

# Chybové situácie



Budeme rozlišovať tieto chybové situácie:

- E1 - neplatný prepínač
- E2 - chýbajúci povinný parameter prepínača
- E3 - parameter prepínača  $-r$  alebo  $-R$  nemá platnú dĺžku

Program vypíše na stdout hlásenie „E3“ a korektne skončí.

# Chybové situácie



Budeme rozlišovať tieto chybové situácie:

- E1 - neplatný prepínač
- E2 - chýbajúci povinný parameter prepínača
- E3 - parameter prepínača  $-r$  alebo  $-R$  nemá platnú dĺžku

Dĺžka parametra  
je 0.

z3.exe -r "" ahoj



# Chybové situácie



Budeme rozlišovať tieto chybové situácie:

- E1 - neplatný prepínač
- E2 - chýbajúci povinný parameter prepínača
- E3 - parameter prepínača *-r* alebo *-R* nemá platnú dĺžku

Dĺžka parametra  
je 23.

```
z3.exe -r "" ahoj
```



```
z3.exe -R totojepriveľmidľheslovo
```

# Bodovanie

Testovacie scenáre		
Scenár 1	Chybové situácie E1, E2 a E3	0,5 b
Scenár 2	Spustenie bez prepínačov	0,5 b
Scenár 3	Prepínač -d	0,5 b
Scenár 4	Prepínač -p	0,5 b
Scenár 5	Prepínač -r	1,5 b
Scenár 6	Prepínač -R	2,0 b
Scenár 7	Prepínač -s	1,0 b
Scenár 8	Prepínač -s (spolu s prepínačom -r alebo -R)	1,5 b
Scenár 9	Kombinácie rôznych prepínačov	2,0 b
Súčet		10 b



Živá ukážka

# Literatúra

- <https://man7.org/linux/man-pages/man3/getopt.3.html>
- <https://slovník.juls.savba.sk/>
- <https://korektor.sk/editor/info>
- <https://app.diagrams.net/>



# Zdroje obrázků

- <https://www.flaticon.com>
- <https://www.freepik.com>