

PROG2

Zadanie 3

Pavol Marák

22. 3. 2022



Obsah 1/2

- Podmienky
- Textový procesor
 - CMD argumenty, getopt
 - Načítavanie riadkov textu
 - Algoritmus spracovania riadku
 - Ukončenie programu
 - Chybové situácie

Obsah 2/2

- Vstup do programu
- Operácie spracovania riadkov textu
- Výstup programu
- Bodovanie

Podmienky

- **Deadline:** 1. apr 2022, 23:59:59
- 10 bodov

Odovzdávanie

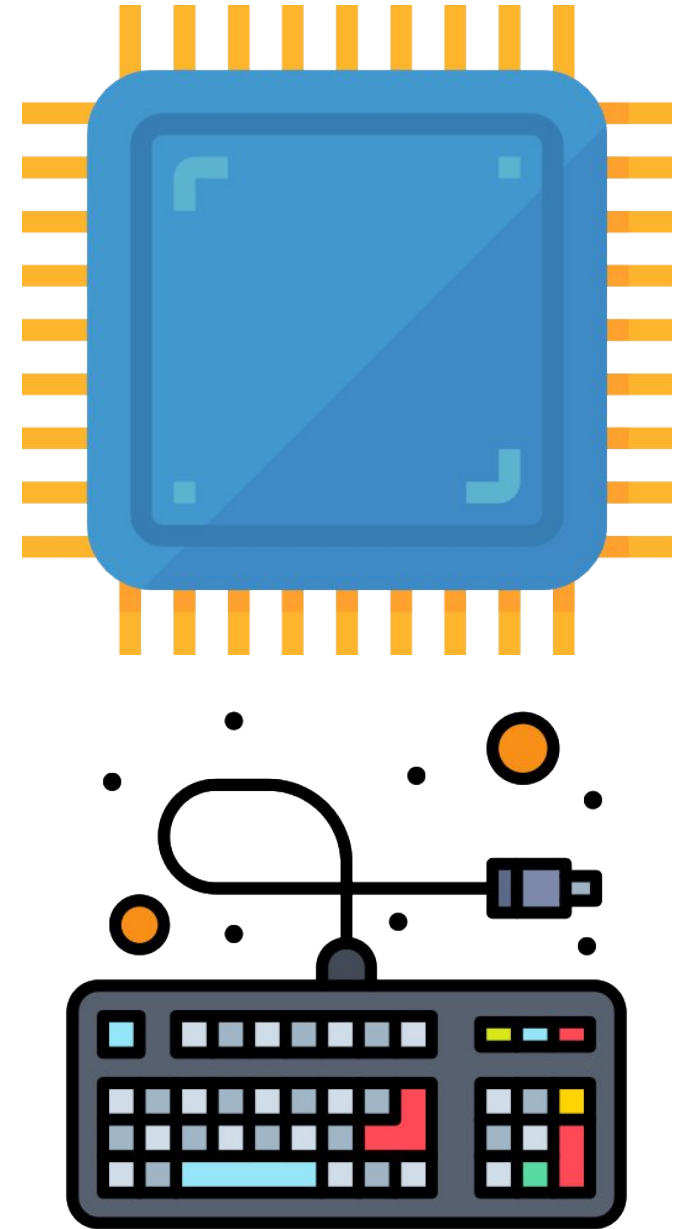
- www.prog2.dev
- kontrola anti-plagiátorským systémom

Penalizácia pri odovzdávaní zadania 3

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

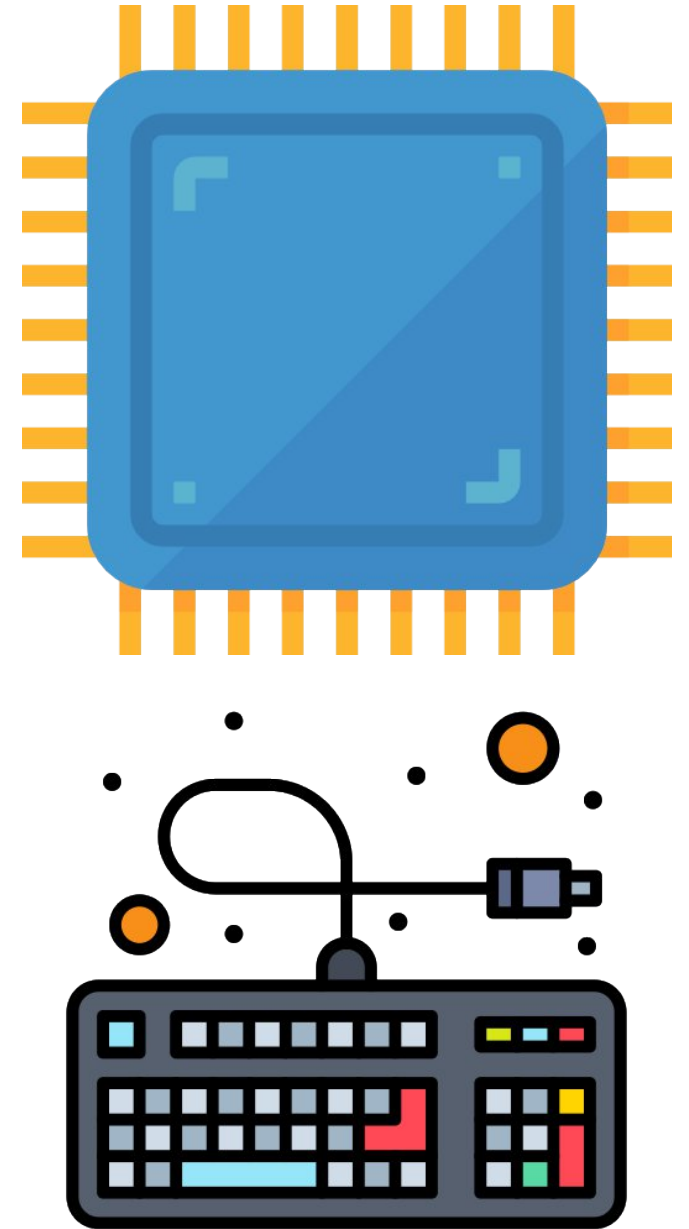
Zadanie 3

- Napíšte terminálový textový procesor v jazyku C, ktorý bude načítavať, spracovávať a vypisovať riadky textu.



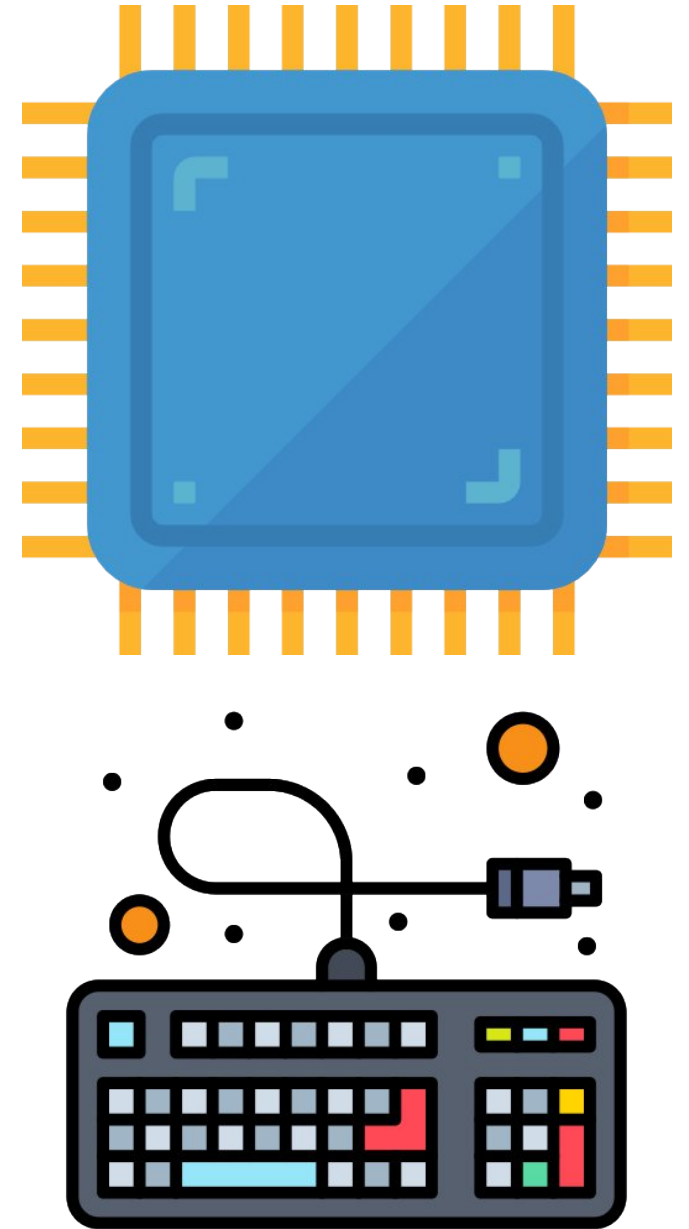
Zadanie 3

- Napíšte terminálový textový procesor v jazyku C, ktorý bude načítavať, spracovávať a vypisovať riadky textu.
- Činnosť programu bude určená command-line argumentami.



Zadanie 3

- Napíšte terminálový textový procesor v jazyku C, ktorý bude načítavať, spracovávať a vypisovať riadky textu.
- Činnosť programu bude určená command-line argumentami.
- Cieľ je precvičiť si prácu s reťazcami.



Zadanie 3

- Zadané CMD argumenty určia aké operácie spracovania riadku sa vykonajú.
- Text bude načítavaný až pokiaľ používateľ nezadá prázdny riadok.

CMD argumenty

- CMD argumenty sú argumenty programu, ktoré sa zadávajú v termináli pri jeho spustení.
- Slúžia na ovládanie správania sa programu z miesta jeho spustenia.
- Každý CMD argument je reprezentovaný ako C reťazec.
- Všetky CMD argumenty sú k dispozícii prostredníctvom poľa reťazcov *argv*, ktoré má dĺžku *argc* prvkov.

CMD argumenty

- Prvok *argv[0]* obsahuje cestu k spúšťanému programu.
- Ostatné prvky poľa *argv* predstavujú používateľom zadané CMD argumenty.
- Pole *argv* a jeho dĺžka *argc* sú parametrami hlavnej funkcie *main*.

CMD argumenty

Príklad spustenia programu s CMD argumentami v Linux termináli:

```
[user@localhost] $ | ./program 123 abc "hello world"
```



argv[0]

Cesta k programu

CMD argumenty

Príklad spustenia programu s CMD argumentami v Linux termináli:

```
[user@localhost] $ | ./program 123 abc "hello world"
```

argv[0]

Cesta k programu

argv[1]

argv[2]

argv[3]

Ostatné argumenty

CMD argumenty

Príklad spustenia programu s CMD argumentami v Linux termináli:

```
[user@localhost] $ | ./program 123 abc "hello world"
```



argv[3]



Pozor: jedná sa o 1 argument
nakoľko je uzavretý v úvodzovkách.

CMD argumenty

Prístup k CMD argumentom vo funkcii *main*.

```
int main(int argc, char* argv[])
{
    /*
        argc - pocet CMD argumentov
        argv - pole argumentov (technicky je to pole retazcov)
    */
    return 0;
}
```


CMD argumenty

V zadání budeme rozlišovat 3 typy CMD argumentov:

- Prepínače (z angl. options)
- Parametre prepínačov
- Non-option argumenty

CMD argumenty

Prepínače

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

CMD argumenty

Parametre prepínačov

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

CMD argumenty

Non-option argumenty

- Každý CMD argument, ktorý nepatrí ani do jednej z dvoch vyššie uvedených kategórií je považovaný za tzv. non-option argument.
- V našom prípade budú všetky non-option argumenty umiestnené až za všetkými prepínačmi a ich parametrami.

CMD argumenty

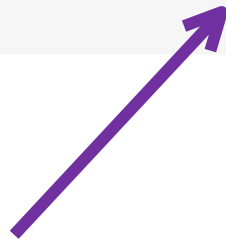
Ukážka rôznych typov CMD argumentov:

```
[user@localhost] $ | ./program -a -b param arg1 arg2 arg3
```

CMD argumenty

Ukážka rôznych typov CMD argumentov:

```
[user@localhost] $ | ./program -a -b param arg1 arg2 arg3
```



Prepínač *-a*

CMD argumenty

Ukážka rôznych typov CMD argumentov:

```
[user@localhost] $ | ./program -a -b param arg1 arg2 arg3
```

Prepínač *-a*

Prepínač *-b* s
parametrom *param*

CMD argumenty

Ukážka rôznych typov CMD argumentov:

```
[user@localhost] $ | ./program -a -b param arg1 arg2 arg3
```

Prepínač *-a*

Prepínač *-b* s
parametrom *param*

Non-option argumenty

getopt

- Proces rozpoznania CMD argumentov sa dá automatizovať pomocou knižničnej funkcie *getopt*.
- Funkcia *getopt* okrem iného odhaľuje aj situácie nesprávneho použitia prepínačov.
- Funkcia *getopt* je dostupná len v Linux prostredí prostredníctvom hlavičkového súboru *unistd.h*.
- Používatelia Windows musia mať nainštalované vhodné prostredie: MinGW, Cygwin, MSYS2 alebo WSL.

getopt

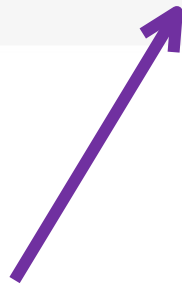
Hlavička funkcie *getopt*:

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

getopt

Hlavička funkcie *getopt*:

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



Počet CMD
argumentov

getopt

Hlavička funkcie *getopt*:

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



Počet CMD
argumentov



Pole CMD
argumentov

getopt

Hlavička funkcie *getopt*:


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



Počet CMD
argumentov



Pole CMD
argumentov



Reťazec špecifikujúci platné
prepínače a ich parametre

getopt

Hlavička funkcie *getopt*:

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



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

Príklad reťazca *optstring*

Reťazec špecifikujúci platné
prepínače a ich parametre

Načítavanie riadkov textu

- Po spustení programu a rozpoznaní CMD argumentov začne textový procesor načítavať riadky textu.
- Na načítanie riadku odporúčame použiť knižničnú funkciu *fgets*.
- V zadaní bude platiť, že **maximálna dĺžka načítaného riadku je 1 000 znakov** (znaky riadku + znak nového riadku '\n').

Načítavanie riadkov textu

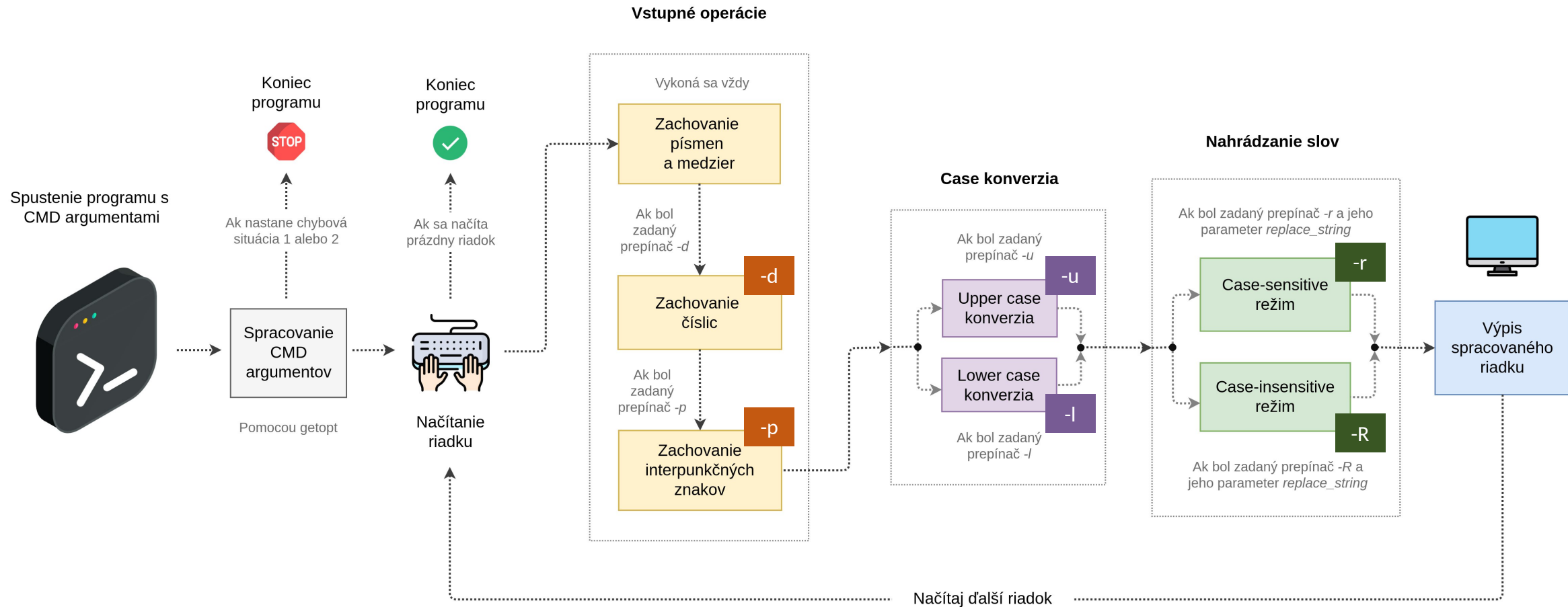
- Načítavanie riadkov bude prebiehať až dovtedy, pokiaľ používateľ nezadá prázdny riadok.
- Prázdny riadok je riadok, ktorý neobsahuje žiadne znaky (t.j. používateľ nič nenapíše, iba stlačí Enter).
- V prípade funkcie *fgets* sa prázdny riadok prejaví ako načítaný reťazec, ktorý bude obsahovať len znak '\n'.

Načítavanie riadkov textu

- Textový procesor po načítaní prázdneho riadku regulérne ukončí svoju činnosť a funkcia *main* vráti hodnotu 0.

Algoritmus spracovania načítaného riadku

- **Poradie operácií spracovania riadku je pevne určené a nemení sa.**
- Zadané prepínače pri spustení programu len určujú, ktorá z operácií sa vykoná.



Ukončenie programu

- Textový procesor končí svoju činnosť po načítaní prázdneho riadku.
- Funkcia *main* vtedy vráti hodnotu 0.

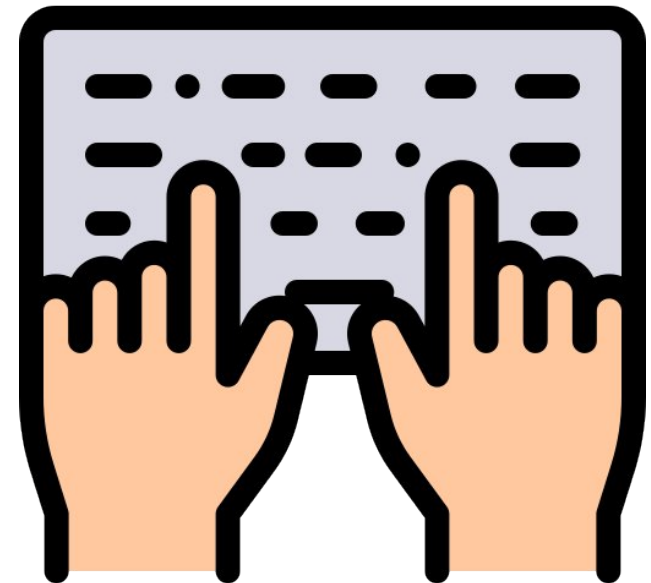
Chybové situácie

Textový procesor môže naraziť na 2 chybové situácie:

- **Situácia 1.** Nastáva ak je program spustený s neplatným prepínačom (nie je v zozname známych prepínačov). Vtedy program musí skončiť bez akéhokoľvek výstupu a funkcia *main* vráti hodnotu 1.
- **Situácia 2.** Nastáva ak nie je zadáný povinný parameter prepínača *-r* alebo *-R*. Vtedy program musí skončiť bez akéhokoľvek výstupu a funkcia *main* vráti hodnotu 2.

Vstup

- Použivatel' bude z klavesnice zadavat' l'ubovol'ne riadky textu s maximálnou dĺžkou 1 000 znakov (znaky riadku + '\n').
- Program skončí po načítaní prázdneho riadku (funkcia *main* vráti hodnotu 0).



Operácie spracovania riadkov textu

1. Vstupné operácie

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

1. Vstupné operácie

Predvolená vstupná operácia

- Po načítaní vstupného riadku sa v ňom ponechajú len malé a veľké písmená anglickej abecedy a medzery.
- Ostatné znaky sa odfiltrujú.
- Táto operácia sa vykoná vždy. Netreba ju špecifikovať žiadnym prepínačom.

1. Vstupné operácie

Prepínač *-d*

- Pri zadaní tohto prepínača sa vo vstupnom riadku okrem predvolených znakov (písmená a medzery) ponechajú aj číslice.
- Na overenie, či je znak číslicou môžeme použiť knižničnú funkciu *isdigit*.

1. Vstupné operácie

Prepínač *-p*

- Pri zadaní tohto prepínača sa vo vstupnom riadku okrem predvolených znakov (písmená a medzery) ponechajú aj interpunkčné znaky.
- Na overenie, či sa jedná o interpunkčný znak môžeme použiť knižničnú funkciu *ispunct*.

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

1. Vstupné operácie

Príklad predvolenej vstupnej operácie

```
1  | ./program
2  | !123Dnes+svieti.slnko456slovo slovo slovo?
3  | Dnessvietislnkoslovo slovo slovo
```

1. Vstupné operácie

Príklad prepínača *-d*

```
1 | ./program -d
2 | Pismena++a ***CISLA 123456 BUDU zachovane. T./e#s]t 96
3 | Pismenaa CISLA 123456 BUDU zachovane Test 96
```

1. Vstupné operácie

Príklad prepínača *-p*

```
1 | ./program -p
2 | ---Interpunkcia+++ sa ne49o612dstran9i. <<<123456789>>> a+r=c
3 | ---Interpunkcia+++ sa neodstrani. <<<>>> a+r=c
```

1. Vstupné operácie

Príklad kombinácie prepínačov *-d* a *-p*

```
1 | ./program -d -p
2 | C9i,s1452a aj [i]nt95erp3unkci\ a nez;;'96aniknu a 9 + P
3 | C9i,s1452a aj [i]nt95erp3unkci\ a nez;;'96aniknu a 9 + P
```

2. Case konverzie

- Tieto operácie slúžia na konverziu načítaných písmen v riadku z veľkých na malé alebo opačne.
- Konverzie sa nesmú kombinovať (testovací softvér ani nebude testovať prípady kombinácie oboch operácií súčasne).

2. Case konverzie

Lowercase konverzia

- Vykoná sa len v prípade, že bol zadany prepínač -l.
- Výsledkom lowercase konverzie je prevod všetkých veľkých písmen na malé. Ostatné znaky riadku zostanú nezmenené.

2. Case konverzie

Príklad lowercase konverzie 1

```
1 | ./program -l
2 | VELKE pismena 123. StRiEdAvE pIsMeNa_?      ABC--1--def
3 | velke pismena  striedave pismena      abcdef
```

2. Case konverzie

Príklad lowercase konverzie 2

```
1 | ./program -d -p -l
2 | ***162AbCdeF PPPPP.W.+U+X    7Aa.
3 | ***162abcdef pppppp.w.+u+x    7aa.
```

2. Case konverzie

Uppercase konverzia

- Vykoná sa len v prípade, že bol zadany prepínač *-u*.
- Výsledkom uppercase konverzie je prevod všetkých malých písmen na veľké. Ostatné znaky riadku zostanú nezmenené.

2. Case konverzie

Príklad uppercase konverzie 1

```
1 | ./program -u
2 | 987 .-]AakfpWRsqb   aaa   TTT   ...p
3 |  AAKFPWRSQB   AAA   TTT   P
```

2. Case konverzie

Príklad uppercase konverzie 2

```
1 | ./program -d -p -u
2 | [[[male slovo]]] 111(((VELKE SLOVO)))999 ... 111 a 9d8T315w
3 | [[[MALE SLOVO]]] 111(((VELKE SLOVO)))999 ... 111 A 9D8T315W
```

3. Nahrádzanie slov

- Rozlišujeme **2 pracovné režimy**:
 - Case-sensitive (prepínač *-r*)
 - Case-insensitive (prepínač *-R*)
- Tieto operácie slúžia na **nahradenie určených slov v riadku** pomocou reťazca, ktorý bol zadaný ako parameter prepínača *-r* alebo *-R*.
- Tento parameter budeme označovať ako *replace_string*.
- Prepínače *-r* alebo *-R* sa nesmú kombinovať (testovací softvér ani nebude testovať prípady kombinácie oboch operácií súčasne).

3. Nahrádzanie slov

- Formát spustenia **case-sensitive** režimu

```
[user@localhost] $ | ./z3 -r replace_string arg1 arg2 arg3
```

- Formát spustenia **case-insensitive** režimu

```
[user@localhost] $ | ./z3 -R replace_string arg1 arg2 arg3
```


3. Nahrádzanie slov

- Formát spustenia **case-sensitive** režimu

```
[user@localhost] $ | ./z3 -r replace_string arg1 arg2 arg3
```

- Formát spustenia **case-insensitive** režimu

```
[user@localhost] $ | ./z3 -R replace_string arg1 arg2 arg3
```

Nepovinné
non-option
argumenty



3. Nahrádzanie slov

- Pri oboch režimoch sa očakáva, že používateľ môže pri spustení programu z terminálu zadať aj tzv. non-option argumenty.
- Množinu non-option argumentov budeme v tomto prípade označovať ako N .

3. Nahrádzanie slov

Slovo

- V zmysle tohto zadania budeme slovom označovať ľubovoľnú alfanumerickú postupnosť znakov, ktorá je oddelená od okolitého textu ne-alfanumerickými znakmi.
- Na zistenie, či je konkrétny znak alfanumerický môžete použiť knižničnú funkciu *isalnum*.
- Každý načítaný riadok textu vieme reprezentovať ako množinu slov $W = \{w_1, w_2, w_3 \dots, w_n\}$.

3. Nahradzanie slov

**Príklady reťazcov, ktoré
považujeme za slová**

hello

WORLD

123word

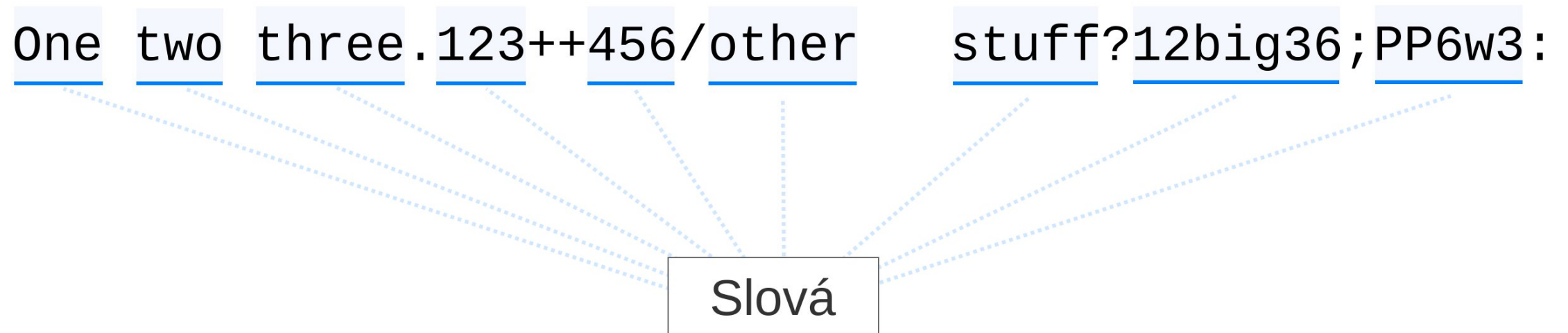
FANTASTIC777

1a2b3C

a

A

4



Príklad načítaného riadku a identifikácia slov, ktoré obsahuje.

3. Case-sensitive režim

```
[user@localhost] $ | ./z3 -r replace_string arg1 arg2 arg3
```

- Počas tejto operácie postupne prechádzame slová v množine W zľava doprava a nahrádzame ich pomocou reťazca *replace_string*.
- Ostatné znaky riadku, ktoré netvoria slová zostávajú nezmenené.

3. Case-sensitive režim



Ktoré slová v načítanom riadku sú nahradené pomocou reťazca *replace_string*?

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*.

3. Case-sensitive režim



Ktoré slová v načítanom riadku sú nahradené pomocou reťazca *replace_string*?

Ak je množina N prázdna.

- V tomto prípade nedochádza k hľadaniu predpony. Namiesto toho sa nahradí každé jedno slovo z množiny *W* pomocou *replace_string*.

3. Case-sensitive režim



Akým spôsobom prebieha nahrádzanie slova v riadku pomocou reťazca *replace_string*?

- Postupne nahrádzame znaky skúmaného slova w_i znakmi reťazca *replace_string*. Musíme rešpektovať veľkosť písmen (z angl. case) v reťazci *replace_string*.
- Ak je *replace_string* kratší ako slovo w_i , nahradíme len príslušnú časť slova w_i a jeho zvyšok ponecháme nezmenený. Ak je *replace_string* dlhší ako slovo w_i , nahrádzanie prebieha len po koniec slova w_i .

3. Case-sensitive režim

Príklad, N je prázdna množina

```
1 | ./program -d -p -r PaPa123
2 | :::Ahoj*** ako sa+dnes.mas?  alfanumericky+programatorsky OK123 ???
3 | :::PaPa*** PaP Pa+PaPa.PaP?  PaPa123ericky+PaPa123atorsky PaPa1 ???
```

3. Case-sensitive režim

Príklad, N je neprázdna množina

```
1 | ./program -d -p -r heSlo123 pri Nie TRI
2 | Priehrada.pri.pri666tomny.priNiest.Nieco.Nieco.niekto.TRI.TRISTO.TRI333
3 | Priehrada.heS.heSlo123mny.heSlo123.heSlo.heSlo.niekto.heS.heSlo1.heSlo1
```

3. Case-insensitive režim

```
[user@localhost] $ | ./z3 -R replace_string arg1 arg2 arg3
```

- Počas tejto operácie postupne prechádzame slová v množine W zľava doprava a nahrádzame ich pomocou reťazca *replace_string*.
- Ostatné znaky riadku, ktoré netvoria slová zostávajú nezmenené.

3. Case-insensitive režim



Ktoré slová v načítanom riadku sú nahradené pomocou reťazca *replace_string*?

Ak je množina N neprázdna.

- Pre každé jedno slovo w_i v množine W hľadáme jeho case-insensitive predponu v množine N .
- Ak pre skúmané slovo w_i nájdeme case-insensitive predponu v množine N , potom dôjde k nahradeniu slova w_i pomocou reťazca *replace_string*.

3. Case-insensitive režim



Ktoré slová v načítanom riadku sú nahradené pomocou reťazca *replace_string*?

Ak je množina N prázdna.

- V tomto prípade nedochádza k hľadaniu predpony. Namiesto toho sa nahradí každé jedno slovo z množiny *W* pomocou *replace_string*.

3. Case-insensitive režim



Akým spôsobom prebieha nahrádzanie slova v riadku pomocou reťazca *replace_string*?

- Postupne nahrádzame znaky skúmaného slova w_i znakmi reťazca *replace_string*.
- Pri alfabetických znakoch reťazca *replace_string* musíme nastaviť veľkosť nahradeného písmena tak, aby korešpondovala s veľkosťou písmena na rovnakej pozícii v slove w_i .

3. Case-insensitive režim



Akým spôsobom prebieha nahrádzanie slova v riadku pomocou reťazca *replace_string*?

- Pri nahrádzaní numerických znakov v slove w_i alfabetickými z reťazca *replace_string*, zachováваме veľkosť písmen z reťazca *replace_string*.
- Ak je *replace_string* kratší ako slovo w_i , nahradíme len príslušnú časť slova w_i a jeho zvyšok ponecháme nezmenený. Ak je *replace_string* dlhší ako slovo w_i , nahrádzanie prebieha len po koniec slova w_i .

3. Case-insensitive režim

Príklad, N je prázdna množina

```
1 | ./program -d -p -R REPlace555
2 | M n SaTurDay hello 123 123456789 rePLace123 BriGhTNESS aAaaaaAa*bXbb.P6p
3 | R r RePlaCe5 repla REP REPlace55 rePLace555 RepLaCE555 rEplacE5*rEp1.REp
```

3. Case-insensitive režim

Príklad, N je neprázdna množina

```
1 | ./program -d -p -R 12Havo sto D Pries 888
2 | 15 pRieStoR priemer dom ST0krat 888 8 Dvere d9r70+Stopka.percent
3 | 15 12haVooR priemer 12h 12Havot 12H 8 12hav 12hav+12havo.percent
```

Výstup

- Každý riadok sa po spracovaní vypíše na štandardný výstup. Okrem spracovaných riadkov sa nevypisuje žiadny iný text.

Bodovanie

Testovacie scenáre		
Scenár 1	Chybová situácia 1.	0.5b
Scenár 2	Chybová situácia 2.	0.5b
Scenár 3	Spustenie programu bez prepínačov.	1.0b
Scenár 4	Len prepínač -d.	0.5b
Scenár 5	Len prepínač -p.	0.5b
Scenár 6	Kombinácia prepínačov -d a -p.	0.5b
Scenár 7	Vstupná operácia + prepínač -l.	0.5b
Scenár 8	Vstupná operácia + prepínač -u.	0.5b

Bodovanie

Scenár 9	Vstupná operácia + prepínač -r. Bez non-option argumentov.	1.0b
Scenár 10	Vstupná operácia + prepínač -r. So zadanými non-option argumentami.	1.0b
Scenár 11	Vstupná operácia + prepínač -R. Bez non-option argumentov.	1.0b
Scenár 12	Vstupná operácia + prepínač -R. So zadanými non-option argumentami.	1.0b
Scenár 13	Rôzne platné kombinácie všetkých prepínačov.	1.5b
Súčet		10 b

Ukážka

Zdroje

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