# Projekt1

**Obsah**



**Legenda:**

Červeně označený text značí změnu zadání. Na změnu zadání bude upozorněno na fóru. Modře označený text upřesňuje zadání. Takto označené fráze nebudou mít na hodnocení téměř žádny vliv.

# Projekt 1 - Práce s textem

## Motivace projektu

Představme si, že máme jednoduchý textový soubor obsahující tabulkové informace (například řádky s jednotlivými buňkami oddělené mezerou). Představme si, že chceme spočítat součet všech hodnot ve třetím sloupci, nebo zjistit jejich aritmetický průměr, odebrat nějaký sloupec nebo přidat nový, který bude představovat součet dvou předchozích buněk? Musíme si spustit těžkopádný tabulkový procesor (Excel, Libreoffice Calc, Google Sheets apod.), data do něj importovat, klikat, co chceme provést, a následně uložit. A co když takových souborů stejného typu, ale s různými daty máme hodně? Hodil by se jednoduchý nástroj pro příkazovou řádku, kterému bychom řekli, co s textovým souborem má dělat, a nástroj automaticky spustili opakovaně nad každým souborem. Nebude to sice hezky "klikatelné", ale ušetří nám to spoustu práce.

## Popis projektu

Cílem projektu je vytvořit program, který bude implementovat základní operace tabulkových procesorů. Vstupem programu budou textová data, zadání operací bude

prostřednictvím argumentů příkazové řádky a svůj výsledek bude program vypisovat na výstup.

## Detailní specifikace

Program implementujte ve zdrojovém souboru *sheet.c*. Vstupní data budou čtena ze standardního vstupu (stdin), výstup bude tisknut na standardní výstup (stdout).

### Překlad a odevzdání zdrojového souboru

Odevzdání: Odevzdejte zdrojový soubor *sheet.c* prostřednictvím informačního systému.

Překlad: Program překládejte s následujícími argumenty

```
$ gcc -std=c99 -Wall -Wextra -Werror sheet.c -o sheet
```

### Syntax spuštění

Program se spouští v následující podobě: (./sheet značí umístění a název programu):

Úprava velikosti tabulky:

```
./sheet [-d DELIM] [Příkazy pro úpravu tabulky]
```

nebo zpracování dat:

```
./sheet [-d DELIM] [Selekce řádků] [Příkaz pro zpracování dat]
```

- Argument `-d` specifikuje, jaké znaky lze interpretovat jako oddělovače jednotlivých buněk. Každý znak ve vstupním řádku, který je obsažen v řetězci DELIM, se chová jako oddělovač dvou sousedících buněk. Ve výchozím nastavení je DELIM řetězec obsahující mezeru. Vícenásobný výskyt stejných znaků v řetězci DELIM je ignorován. První znak z řetězce DELIM bude také použit jako oddělovač výstupních hodnot.

### Úprava tabulky

Úpravy tabulky způsobují zvětšení či zmenšení tabulky, resp. řádků a sloupců. Příkazů pro úpravu tabulky může být zadáno více. V takovém případě budou zadány jako sekvence více argumentů příkazové řádky:

Příkazy pro úpravu tabulky:

- `irow R` - vloží řádek tabulky před řádek R > 0 (insert-row).
- `arow` - přidá nový řádek tabulky na konec tabulky (append-row).
- `drow R` - odstraní řádek číslo R > 0 (delete-row).

- `drows N M` - odstraní řádky N až M (N <= M). V případě N=M se příkaz chová stejně jako `drow N`.
- `icol C` - vloží prázdný sloupec před sloupec daný číslem C.
- `acol` - přidá prázdný sloupec za poslední sloupec.
- `dcol C` - odstraní sloupec číslo C.
- `dcols N M` - odstraní sloupce N až M (N <= M). V případě N=M se příkaz chová stejně jako `dcol N`.

## Zpracování dat

Zpracování dat představuje úpravy obsahu jednotlivých buněk tabulky. Příkazy pro zpracování dat jsou následující:

- Příkazy, které jsou povinné pro úspěšné splnění projektu:
  - `cset C STR` - do buňky ve sloupci C bude nastaven řetězec STR.
  - `tolower C` - řetězec ve sloupci C bude převeden na malá písmena.
  - `toupper C` - řetězec ve sloupce C bude převeden na velká písmena.
  - `round C` - ve sloupci C zaokrouhlí číslo na celé číslo.
  - `int C` - odstraní desetinnou část čísla ve sloupci C.
  - `copy N M` - přepíše obsah buněk ve sloupci M hodnotami ze sloupce N.
  - `swap N M` - zamění hodnoty buněk ve sloupcích N a M.
  - `move N M` - přesune sloupec N před sloupec M.
- Příkazy, které jsou volitelné pro úspěšné splnění projektu:
  - `csum C N M` - do buňky ve sloupci C bude uloženo číslo reprezentující součet hodnot buněk na stejném řádku ve sloupcích N až M včetně (N <= M, C nesmí patřit do intervalu <N;M>).
  - `cavg C N M` - obdobně jako csum, avšak výsledná hodnota představuje aritmetický průměr hodnot.
  - `cmin C N M` - obdobně jako csum, avšak výsledná hodnota představuje nejmenší nalezenou hodnotu.
  - `cmax C N M` - obdobně jako cmin, jedná se však o maximální nalezenou hodnotu.
  - `ccount C N M` - obdobně jako csum, avšak výsledná hodnota představuje počet neprázdnných hodnot daných buněk.
  - `cseq N M B` - do buněk ve sloupcích N až M včetně vloží postupně rostoucí čísla (o jedničku) počínaje hodnotou B.
  - `rseq C N M B` - ve sloupci C do buněk každého řádku od řádku N po řádek M včetně vloží rostoucí čísla počínaje hodnotou B. Číslo M může být nahrazeno pomlčkou. V takovém případě se tím myslí poslední řádek souboru.

- `rsum C N M` - do buňky ve sloupci C na řádku M+1 vloží součet hodnot buněk ve sloupci C na řádcích N až M včetně.
- `ravg C N M` - obdobně jako rsum, avšak výsledná hodnota představuje aritmetický průměr.
- `rmin C N M` - obdobně jako rsum, avšak výsledná hodnota představuje nejmenší hodnotu.
- `rmax C N M` - obdobně jako rsum, avšak výsledná hodnota představuje největší hodnotu.
- `rcount C N M` - obdobně jako rsum, avšak výsledná hodnota představuje počet neprázdnných hodnot daných buněk.

## Selekce řádků

Příkazy pro zpracování dat mohou být aplikovány nejen na celé tabulce, ale pouze na vybraných řádcích. Příkazy selekce takových řádků budou zadány na příkazovou řádku před příkazy pro zpracování dat:

- `rows N M` - procesor bude zpracovávat pouze řádky N až M včetně (N <= M). N=1 znamená zpracování od prvního řádku. Pokud je místo čísla M zadán znak - (pomlčka), ta reprezentuje poslední řádek vstupního souboru. Pokud je pomlčka také místo sloupce N, myslí se tím výběr pouze posledního řádku. Pokud není tento příkaz zadán, uvažuje se implicitně o všech řádcích.
- `beginswith C STR` - procesor bude zpracovávat pouze ty řádky, jejichž obsah buňky ve sloupci C začíná řetězcem STR.
- `contains C STR` - procesor bude zpracovávat pouze ty řádky, jejichž buňky ve sloupci C obsahují řetězec STR.

Bude kontrolován maximálně jeden příkaz pro selekci řádků. Kombinace selekce řádků zadání nedefinuje a ponechává na implementaci.

## Implementační detaily

- Výslednou tabulku program vypíše na výstup.
- Vstupní tabulka nesmí být prázdný soubor.
- Žádné číslo příkazu identifikující řádek nebo sloupec nesmí být menší než 1.
- Program předpokládá, že počet buněk v prvním řádku tabulky reprezentuje počet sloupců celé tabulky. Pokud bude na vstupu tabulka nesplňující této podmínky, je chování programu nedefinované (program však nesmí skončit chybou za běhu, např. chyba přístupu do paměti).
- V případě, že tabulka neobsahuje počet řádků nebo sloupců požadovené v zadaných příkazech, je tato skutečnost ignorována: (i) příkazy upravující tabulku nebudou na chybějících řádcích nebo sloupcích provedeny, (ii) kritéria výběru dat nad chybějícími řádky nebo sloupci nebudou splněna.

- V případě kombinace selekce řádků s příkazy `rseq` až `rcount` se nebudou započítavat data těch řádků, které neodpovídají dané selekci (např. pro aritmetický průměr se budou započítávat pouze ty řádky, které odpovídají zadanému kritériu).
- Maximální podporovaná délka řetězce v buňce nebo argumentu je 100. Maximální délka celého řádku je 10KiB. Při delších řetězcích program upozorňuje chybovým hlášením a ukončením s chybovým kódem.

### Omezení v projektu

Je zakázané použít následující funkce:

- volání z rodiny malloc a free - práce s dynamickou pamětí není v tomto projektu zapotřebí,
- volání z rodiny fopen, fclose, fscanf, ... - práce se soubory (dočasnými) není v tomto projektu žádoucí,
- volání qsort, lsearch, bsearch a hsearch - cílem je zamyslet se nad algoritmizací a strukturou dat.
- volání funkce exit - cílem projektu je naučit se vytvořit programové konstrukce, které dokáží zpracovat neočekávaný stav programu.

## Neočekávané chování

Na chyby za běhu programu reagujte obvyklým způsobem: Na neočekávaná vstupní data, formát vstupních dat nebo chyby při volání funkcí reagujte přerušením programu se stručným a výstižným chybovým hlášením na příslušný výstup a odpovídajícím návratovým kódem. Hlášení budou v kódování ASCII česky nebo anglicky.

## Příklady vstupů a výstupů

Pomocný soubor telefonního seznamu:

```
$ cat tab1.txt
Cviceni:Bodu:Projekt
Prvni programy:0:
Cyklus, typy:0:
Retezce:0:
Funkce 1:0:
Struktury:2:
Ukazatele:2:
Funkce:0:Obhajoba 1
Malloc, dbg:2:
:0:
:0:
Iterace, rekurze:2:
:0:Obhajoba 2
Dynamicke struktury:2:
$ ./sheet -d : icol 1 <tab1.txt >tab1a.txt
$ ./sheet -d : rows 1 1 cset 1 Tyden <tab1a.txt >tab2.txt
$ cat tab2.txt
```

```
Tyden:Cviceni:Bodu:Projekt
:Prvni programy:0:
:Cyklus, typy:0:
:Retezce:0:
:Funkce 1:0:
:Struktury:2:
:Ukazatele:2:
:Funkce:0:Obhajoba 1
:Malloc, dbg:2:
::0:
::0:
:Iterace, rekurze:2:
::0:Obhajoba 2
:Dynamicke struktury:2:
$ ./sheet -d : rseq 1 2 - 1 <tab2.txt >tab3.txt
$ cat tab3.txt
Tyden:Cviceni:Bodu:Projekt
1:Prvni programy:0:
2:Cyklus, typy:0:
3:Retezce:0:
4:Funkce 1:0:
5:Struktury:2:
6:Ukazatele:2:
7:Funkce:0:Obhajoba 1
8:Malloc, dbg:2:
9::0:
10::0:
11:Iterace, rekurze:2:
12::0:Obhajoba 2
13:Dynamicke struktury:2:
$ ./sheet -d : arow <tab3.txt >tab3a.txt
$ ./sheet -d : rows - - cset 2 "celkem bodu" <tab3a.txt >tab4.txt
$ cat tab4.txt
Tyden:Cviceni:Bodu:Projekt
1:Prvni programy:0:
2:Cyklus, typy:0:
3:Retezce:0:
4:Funkce 1:0:
5:Struktury:2:
6:Ukazatele:2:
7:Funkce:0:Obhajoba 1
8:Malloc, dbg:2:
9::0:
10::0:
11:Iterace, rekurze:2:
12::0:Obhajoba 2
13:Dynamicke struktury:2:
:celkem bodu::
$ ./sheet -d : rsum 3 2 14 <tab4.txt >tab5.txt
$ cat tab5.txt
Tyden:Cviceni:Bodu:Projekt
1:Prvni programy:0:
2:Cyklus, typy:0:
```

```
3:Retezce:0:
4:Funkce 1:0:
5:Struktury:2:
6:Ukazatele:2:
7:Funkce:0:Obhajoba 1
8:Malloc, dbg:2:
9::0:
10::0:
11:Iterace, rekurze:2:
12::0:Obhajoba 2
13:Dynamicke struktury:2:
:celkem bodu:10:
```

## Hodnocení

Na výsledném hodnocení mají hlavní vliv následující faktory:

- přeložitelnost zdrojového souboru,
- formát zdrojového souboru (členění, zarovnání, komentáře, vhodně zvolené identifikátory),
- dekompozice problému na podproblémy (vhodné funkce, vhodná délka funkcí a parametry funkcí),
- správná volba datových typů, případně tvorba nových typů,
- správná funkcionalita úprav a zpracování tabulky a
- ošetření chybových stavů.

## Priority funkcionality

1. Příkazy úpravy tabulky (každé spuštění programu akceptuje maximálně jeden příkaz).
2. Povinné příkazy zpracování dat.
3. Sekvence příkazů úprav tabulky (každé spuštění programů akceptuje i více příkazů úprav tabulky ve více argumentech).
4. Příkazy pro selekci řádků.
5. Vybrané volitelné příkazy zpracování dat. Hodnocení této implementace bude nad rámec maximálního hodnocení projektu a bude uznáno pouze v případě bezproblémové implementace povinných částí.
6. Další prémiové řešení (viz níže).

## Prémiové řešení

Prémiové řešení je dobrovolné a lze za něj získat bonusové body. Podmínkou pro udělení prémiových bodů je výborné vypracování povinných částí projektu zadání. Výsledné hodnocení je plně v kompetenci vyučujího, který bude projekt hodnotit. Výše prémiových bodů závisí také na sofistikovanosti řešení. Příklady prémiového řešení mohou být:

- Implementace volitelné identifikace sloupců a rozsahů ve stylu pokročilých tabulkových procesorů - pomocí písmen: sloupec 1 je označen písmenem A, rozsah sloupců 26-28 řetězcem Z-AB. Kombinace rows a cols jako příkaz `range A1:S42` (toto je pouze příklad).
- Implementace příkazu `concatenate N M STR`, který spojí řetězce všech buněk ve sloupcích N až M spojovacím řetězcem STR, přičemž výsledek uloží do sloupce N a odstraní všechny ostatní sloupce až do sloupce M. Řetězec STR může být prázdný.
- Implementace příkazu `split N D`, který řetězec v buňce ve sloupci N rozdělí podle oddělovače D (jeden znak) na několik sousedících sloupců počínaje N. Všechny ostatní sloupce budou odsunuty doprava. Záleží na implementaci, jak program naloží s proměnným počtem sloupců v různých řádcích.