Formát textu

Rozmanitost informací, které se mají promítnout na výstupní stránku, je poměrně značná na takto malou úlohu, uděláme proto nejlépe, když si úlohu nejprve koncepčně co nejvíce zjednodušíme. Vyjdeme od vhodného návrhu datových struktur.

1. Výstupní otisk. Pro něj si rezervujeme dvourozměrné pole znaků, které bude přesně obsahovat výstupní otisk a které v závěru jen vytiskneme bez jakýchkoli dalších úprav.

2. Výstupní otisk je rozdělen na textové sloupce a řádky. Sloupců se hned na začátku zbavíme tak, že si ukaždého řádku poznamenáme, v kterém sloupci výstupního rastru začíná a jakou má kapacitu, to jest jaký počet znaků ve výstupním otisku má k dispozici (skutečný text, který se později v řádku objeví, tuto kapacitu nemusí celou využít). V implementaci to provedeme tak, že všechny řádky očíslujeme shora dolů a odleva doprava. Každý řádek, bez ohledu na to, v kterém leží sloupci, bude mít své unikátní pořadové číslo. Jakmile načteme počet textových sloupců, vytvoříme dvě celočíselná pole indexovaná čísly řádků a do nich zapíšeme pro každý řádek jeho počáteční rastrový sloupec (= horizontální souřadnici) a jeho kapacitu podle šířky příslušného sloupce.. Kromě toho analogicky vytvoříme ještě jedno pole, které bude registrovat vertikální souřadnice řádků ve výstupním otisku.
Pro každý řádek tak budeme sledovat tři veličiny horizontální a vertikální souřadnici jeho začátku a jeho kapacitu. Alternativně bychom mohli registrovat nikoli kapacitu, ale horizontální souřadnici posledního možného znaku na řádku, koncepčně v tom není rozdíl.
Součástí informace o řádku nebude text, který se objeví na výstupním otisku. Text budeme později přenášet rovnou ze vstupního textu na výstupní otisk. Informace o řádku budou tento přenos jenom usměrňovat, uvedená tři pole k tomu postačí. Vyhneme se tak duplikaci částí textu, čímž nám ubyde i starosti o jejich správu a tím (snad ?) i o případné další zavlečené chyby v kódu.

3. Obrázky
Každá obrázek, jakmile načteme jeho parametry, zakreslíme do pole výstupního otisku. Nebudeme tedy ani pro obrázky rezervovat žádné datové struktury. Obrázek může celkově nebo částečně překrývat některé řádky textu ve výstupním otisku. Horizontální souřadnice a/nebo kapacitu takových řádků je tedy nutno přizpůsobit tomuto omezení. Pro každý obrázek proto projdeme všechny řádky, to jest zároveň pole horizontálních souřadnic a pole kapacit, a upravíme jejich hodnoty, podle toho, zda a jak řádek koliduje s obrázkem.
Po přečtení všech obrázků mohou mít některé řádky na výstupním otisku nulovou kapacitu,
těch se zbavíme jednoduše tak, že hodnoty ve všech polích „setřeseme“ směrem k začátku podle toho, zda je příslušná kapacita nenulová, a snížíme hodnotu celkového počtu řádků.

4. Vstupní text
Vstupní text načteme do jediného pole vstupních znaků a již při načítání přeskočíme všechny duplicitní nebo nadbytečné mezery a konce neprázdných řádků nahradíme případně mezerou. Každý prázdný řádek indikujeme v poli vstupních znaků jediným speciálním znakem. Takto připravený text již bude snazší dělit na jednotlivé úseky odpovídající kapacitám řádků ve výstupním otisku. Načtení textu provede metoda `readInputText`, jejíž činnost je nejlépe patrná ze samotného kódu (s komentáři).

5. Zarovnání textu
Parametr zarovnání textu pouze přečteme ze vstupního souboru a během dělení vstupního textu do jednotlivých řádků ve výstupním otisku se jím nebudeme zabývat.

6. Dělení vstupního textu do jednotlivých řádků
Tato část úlohy je patrně nejméně přehledná. Podle zadání úlohy je nutno sestavit vlastní originální algoritmus, který bere v úvahu všechny požadavky. Částečného a užitečného zjednodušení je dosaženo pomocí redukce v bodě 4. Kupodivu se ukazuje, že k vyřešení tohoto bodu stačí tři indexy a jeden průchod vstupním textem. První index registruje pozici ve vstupním textu, od níž se začne plnit aktuální výstupní řádek. Druhý index registruje pozici začátku aktuálně procházeného slova ve vstupním textu. O tomto slově je nutno rozhodnout, zda se celé vejde na aktuální výstupní řádek nebo zda se objeví až na dalším

řádku nebo zda jej bude nutno dělit. Třetí index registruje aktuální čtenou pozici ve vstupním textu. Před každým posunem třetího indexu je nutno ve vhodném pořadí zkontrolovat, zda je aktuální slovo dočteno, zda je naplněna kapacita řádku, zda se slovo dělí do více výstupních řádků a podle toho indexy pohybovat. Každé slovo se snažíme přenést do výstupního řádku co nejdříve, jakmile je přečteme celé nebo jeho maximální přenesitelnou část. Detailní záznam celého postupu je v kódu ve funkci `transfer`, je při něm nutno dávat pozor na pečlivou manipulaci s indexy.

7. Přenášení částí vstupního textu do řádků na výstupním otisku je po zvládnutí bodu 6 již snadné. Indexy určují, o jakou část textu se jedná, nadbytečné mezery byly odstraněny v bodu 4 a zbývá jen otázka zarovnání. Kapacitu řádku známe, objem přenášeného textu rovněž známe, zbývá jej tedy pouze doplnit zleva nebo zprava nebo z obou stran mezerami na plnou kapacitu řádku.

8. Nezmínili jsme rámec otisku, ten se však ihned po načtení jeho parametrů zakreslí do výstupního otisku podobně jako obrázky, pro algoritmus řešení úlohy není významný.