Programiranje 1 — osma domača naloga

Rok za oddajo: nedelja, 5. januarja 2020, ob 23:55

Podatkovni tipi

Naloga

Napišite program, ki prebere zaporedje deklaracij podatkovnih tipov v izmišljenem programskem jeziku, nato pa po vrsti izvrši ukaze, od katerih vsak zahteva izpis *opisa*, *velikosti* ali *pomnilniške porabe* enega od deklariranih tipov.

Opis tipa je niz, ki podaja vrsto in zgradbo tipa. *Velikost* tipa je število bajtov, ki jih potrebujemo za predstavitev spremenljivk tega tipa. *Pomnilniška poraba* tipa pa je število bajtov, ki jih spremenljivka tega tipa zaseda v pomnilniku. Pomnilniško porabo *katerega-koli* tipa izračunamo tako, da njegovo velikost zaokrožimo navzgor na najbližji večkratnik velikosti *pomnilniške besede* v bajtih (B). Na primer, če velja $B = 8$, potem vse spremenljivke tipov velikosti do vključno 8 bajtov (torej tudi tipov z velikostjo 1 bajt) zasedajo po 8 bajtov pomnilnika. Razlog za takšno razsipnost je v večji učinkovitosti obdelave podatkov, saj večina procesorskih ukazov deluje nad pomnilniškimi besedami, ne nad posameznimi bajti.

Vsak tip, ki je deklariran na vhodu, pripada eni od sledečih vrst:

Primitivni tip z določeno velikostjo. Primitivni tip je tip, ki ga ni mogoče deliti na manjše enote. Deklariran je tako:

`prim` v

Velikost tako deklariranega tipa znaša kar v, njegov opis pa je tak:

`prim`v

Na primer, velikost tipa z deklaracijo

`prim 3`

znaša 3, njegova pomnilniška poraba znaša $z_B(3)$ ($z_B(.)$ označuje zaokrožitev navzgor na najbližji večkratnik števila B), njegov opis pa je niz

`prim3`

Tabela. Tabela je tip, sestavljen iz n elementov istega tipa. Deklarirana je kot

`arr` n t

pri čemer je t tip posameznega elementa tabele. Velikost tabele je enaka zmnožku števila elementov in velikosti tipa elementa, njen opis pa je niz

opis(t)`[`n`]`

pri čemer je opis(t) opis tipa t. Na primer, deklaracija

`arr 3 prim 7`

podaja tabelo treh elementov primitivnega tipa velikosti 7. Njena velikost je enaka $3 \cdot 7 = 21$, pomnilniška poraba znaša $z_B(21)$, njen opis pa je niz

`prim7[3]`

Navadna struktura. Navadna struktura je tip, sestavljen iz n komponent poljubnih tipov. Deklarirana je kot

`ostruct` $n\ t_1\ t_2\ \ldots\ t_n$

pri čemer so $t_1, \ldots, t_n$ tipi posameznih komponent. Velikost navadne strukture je enaka vsoti *pomnilniških porab* posameznih komponent, njen opis pa je niz

`ostruct{`opis(t_1)`,`␣opis(t_2)`,`␣`...,`␣opis(t_n)`}`

pri čemer je opis(t_i) (za $i \in \{1, \ldots, n\}$) opis tipa t_i. Na primer, deklaracija

```
ostruct 3 prim 1 prim 8 arr 3 prim 4
```

podaja navadno strukturo, sestavljeno iz dveh primitivnih tipov in tabele treh elementov primitivnega tipa. Velikost te strukture znaša $z_B(1) + z_B(8) + z_B(3 \cdot 4)$, pomnilniška poraba je enaka $z_B(z_B(1) + z_B(8) + z_B(3 \cdot 4))$, njen opis pa je niz

```
ostruct{prim1, prim8, prim4[3]}
```

Varčna struktura. Varčne strukture se od navadnih razlikujejo zgolj po uporabi besede `pstruct` namesto `ostruct` (tako v deklaraciji kot v opisu) in v načinu računanja velikosti. Velikost varčne strukture je namreč enaka vsoti *velikosti* posameznih komponent. Na primer, velikost varčne strukture

```
pstruct 3 prim 1 prim 8 arr 3 prim 4
```

znaša $1 + 8 + 3 \cdot 4 = 21$, pomnilniška poraba pa $z_B(21)$.

Vhod

Vsa števila, ki se v kakršnikoli vlogi pojavljajo na vhodu, so cela števila v intervalu $[1, 1000]$.

V prvi vrstici vhoda je podano število B (velikost pomnilniške besede). V drugi vrstici je podano število D. Sledi D vrstic s posameznimi deklaracijami tipov v skladu z gornjimi opisi. V naslednji vrstici je podano število U. Sledi U vrstic s posameznimi ukazi. Vsak ukaz je sestavljen iz dveh števil:

$a\ \ d$

Velja $a \in \{1, 2, 3\}$ in $d \in [1, D]$. Če je $a = 1$, potem ukaz zahteva izpis opisa tipa, deklariranega v d-ti vrstici (v okviru vrstic, ki podajajo deklaracije tipov). Ukaz z vrednostjo $a = 2$ zahteva izpis velikosti, ukaz z vrednostjo $a = 3$ pa izpis pomnilniške porabe tipa, deklariranega v d-ti vrstici.

Skriti testni primeri 1–20 vsebujejo samo deklaracije primitivnih tipov. Primeri 21–30 vsebujejo tudi deklaracije tabel primitivnih tipov. Primeri 31–35 vsebujejo tudi deklaracije navadnih in varčnih struktur, sestavljenih zgolj iz primitivnih tipov. Primeri 36–40 vsebujejo tudi vgnezdene deklaracije tabel (tabela tabel, tabela tabel tabel ...), ne pa tabele struktur. Primeri 41–50 vsebujejo tudi tabele struktur ter strukture, sestavljene iz primitivnih tipov, tabel in struktur, pri čemer je maksimalna globina gnezdenja enaka kvečjemu 100.

Izhod

Izpišite U vrstic z rezultati posameznih ukazov. V i-ti vrstici naj bo izpisan rezultat i-tega ukaza. Ne izpisujte dodatnih presledkov, praznih vrstic itd.! Vsi številski rezultati bodo zanesljivo manjši od 10^9.

Javni testni primer

Vhod:[1]

```
8
6
prim 3
arr 10 prim 5
arr 5 arr 3 arr 3 prim 7
ostruct 3 prim 1 prim 8 prim 4
pstruct 3 prim 1 prim 8 prim 4
pstruct 3 arr 15 ostruct 2 prim 3 prim 7 ↵
          pstruct 1 arr 4 ostruct 2 prim 4 prim 1 ↵
          arr 1 prim 3
18
1 1
2 1
3 1
1 2
2 2
3 2
1 3
2 3
3 3
1 4
2 4
3 4
1 5
2 5
3 5
1 6
2 6
3 6
```

Izhod:

```
prim3
3
8
prim5[10]
50
56
prim7[3][3][5]
315
320
```

[1] Z znakom ↵ sta označena umetna preloma vrstice. Vhodna datoteka teh prelomov ne vsebuje.

```
ostruct{prim1, prim8, prim4}
24
24
pstruct{prim1, prim8, prim4}
13
16
pstruct{ostruct{prim3, prim7}[15], pstruct{ostruct{prim4, prim1}[4]}, prim3[1]}
307
312
```

Oddaja naloge

Svoj program oddajte v obliki datoteke `DN08_vvvvvvvv.java`, pri čemer *vvvvvvvv* nadomestite s svojo vpisno številko. Ker boste po vsej verjetnosti napisali več razredov, jih definirajte kot statične notranje razrede znotraj glavnega razreda `DN08_vvvvvvvv`:

```
public class DN08_vvvvvvvv {
    ...
    public static void main(String[] args) {
        ...
    }

    private static class Razred1 {
        ...
    }

    private static class Razred2 {
        ...
    }
    ...
}
```

Statični notranji razredi se obnašajo enako kot samostojni razredi. Priporočamo vam, da najprej vsak razred napišete v svoji datoteki (kot smo to delali na vajah), nato pa vse razrede preselite v glavni razred ter jih opremite z določiloma `private` in `static`. Po tej operaciji delovanje programa še enkrat preverite!

Napotek

Če je `sc` objekt tipa `Scanner`, potem naslednjo besedo (niz znakov do prvega presledka, tabulatorja ali preloma vrstice) preberete s klicem `sc.next()`.