Eesti koolinoorte informaatika lahtine võistlus

19. oktoober 1996. a.

Kommentaare I vooru ülesannete kohta

Kõigepealt märgime, et vanema ja noorema rühma variantides on üldideede poolest tegemist ühtede ja samade ülesannetega. Vanemal rühmal on ainult vaja läbi programmeerida rohkem detaile, s.t. töö mahu poolest pole variandid võrdsed.

Samuti leidub ülesande 2 noorema rühma versioonil mõistlikum "teoreetiline" tsüklilisuse kriteerium ja ülesandes 3 on nooremal rühmal rohkem lootust saada punkte ka aeglase algoritmiga. Kommentaarid anname siin ülesannete kaupa kahe rühma jaoks koos.

---

1. LIITMINE

Ülesanne on lihtne ja võimaldab lahendamist mitmes erinevas järjekorras. Loomulik on enne arvude ekraanile kirjutamist leida summa ja siis arvude ekraanile kirjutamiseks välja arvutada, milline on maksimaalne kohtade arv enne koma. Aga et ülesandes pole otseselt nõutud väljundi paigutamist ekraani vasakule äärele, võib ka määrata piisava varuga üheliste asukoha ja asuda väljastama, paigutades arvu alguse vajalikule kohale.

Kõige rohkem vigu oli ülekannete juures. Nende leidmiseks tuleb tavaline liitmise algoritm ikkagi ausalt paremalt vasakule läbi teha, muidu tekivad raskematel juhtudel vead. Vanemas rühmas paigutas mõni programm ülekandeid ka koma kohale.

---

2. PROGRAMMEERIMISKEEL

Seda ülesannet on mõlemas rühmas lahendatud kõige edukamalt.

On selge, et tuleb võtta vaadeldava muutuja väärtuse jaoks üks oma programmi muutuja. Sellele muutujale tuleb anda ülesandes määratud algväärtus ja hakata siis programmi tööd läbi mängima, kontrollides iga tehte järel muutuja väärtuse lubatavust ja suundudes vajadusel lähteprogrammi 1. rea täitmisele.

Lahendamist vajab veel lõpmatu tsükli äratundmise probleem. Põhimõtteliselt võib seda teha kas mingi kavala matemaatilise kriteeriumi väljamõtlemise teel või programmi tööd jälgides. Kavala kriteeriumi kontroll võib olla kergem programmee-rida, aga siis peab autor ka veendunud olema, et tema kriteerium tõepoolest täpselt paika peab. Tubli pool lahendustest teatavad mõne testi puhul lõpmatust tsüklist seal, kus teda tegelikult ei ole! Järelikult olete vale ja ülejõu käiva strateegia valinud.

Programmeerija peab oskama vahet teha, kus oma pead vaevata ja kus masinal töötada lasta.

Et programmi tööd nagunii käsukaupa modelleeritakse, on lihtsam jälgida programmi täitmist ja siis teatud momendil otsustada, et on tegemist tsükliga. Sellise otsuse aluseks võib olla näiteks

a) programmi täitmist alustatakse teist korda sama muutuja väärtusega 1. realt (selleks on vaja massiivi, kus registreeritakse, millise muutuja väärtusega on juba 1. realt alustatud),

b) alustatakse juba 1002. korda 1. realt. Siis peab muutuja väärtus mõnda eelmist kordama.

Märgime siinjuures, et programmis oleva tsükli 1000-kordne läbimängimine pole midagi koledat:

1. tehakse kõigest mõned (kümned) tuhanded operatsioonid,

2. nagunii leiduvad meie ülesande jaoks näited, kus seda tuleb teha ja saadakse positiivne vastus.

Mõtlemiseks: leidke maksimaalse korduste arvuga näide, kus tsükkel pole lõpmatu.

---

3. PIKIM SÕNA

Piisavalt kiire algoritm võiks olla järgmine: Moodustame **N**x**N** tabeli, kuhu püüame igasse ruutu saada maksimaalse numbri, mitmendal kohal see täht saab olla moodustatavas sõnas.

Esialgu täidame tabeli arvudega 1.

Edasi kirjutame

1. arvu 2 igasse sellisesse ruutu, kuhu saab mingist ruudust arvuga 1 liikuda lubatud käigu abil,

2. siis arvu 3 igasse sellisesse ruutu, kuhu saab mingist ruudust arvuga 2 liikuda lubatud käigu abil

3. jne, kuni veel tekib uusi arve.

Selle töö tulemusena tekib tabeli mingisse ruutu maksimaalne võimalik sõna pikkus. Tuleb veel väljastada marsruut. Selleks võime ülalkirjeldatud tabeli moodustamise ajal jätta iga ruudu jaoks meelde ka ruudu, millest sinna liiguti, kasutades selleks näiteks kummagi koordinaadi jaoks veel ühte tabelit. Võib aga ka liikuda piki maksimaalset sõna tähthaaval tagasi eelmise numbriga ruutu.

Mõtlemiseks. Kui kirjutasite võistlusel kõiki sõnade moodustamise variante läbi vaatava (tavaliselt rekursiivse) programmi ja see sai N10 korral hakkama, siis kirjutage ka ülaltoodud algoritmi järgi ja võrrelge neid N=15, 20, 25, ... korral.

Ülesandeid kommenteeris Rein Prank