

4. labor kis ZH, 2018. április 26., csütörtök 10-12

NÉV (olvasható):

NEPTUN kód:

max. 10p, megoldási idő: 20 perc

1. Készíts C++ programot, mely a szabványos bemeneten érkező szövegben található hivatkozásokat a szabványos kimenetre írja. A szöveg végét @ karakter jelzi. A megoldáshoz fel kell használnod a C++ string osztályt, az std::getline() függvényt, valamint a string osztály find és substr tagfüggvényét. Pl. Bemenet: „*There are models for the computation of the local heat transfer coefficient [12], and even for non-uniform temperature and for non-uniform heat-flow [14], [15]. This positive feedback can result in S shaped current-voltage characteristics, and if the generated heat cannot be dispersed into the environment, thermal runaway develops that leads to the destruction of the device [8], [16], [17].* @” Kimenet: 12 14 15 8 16 17. A hivatkozások [] között vannak, szögletes zárójel más módon nem szerepel a szövegben. Hivatkozás nem lehet két sorba törve. (max. 6p)

```
#include <iostream>
#include <string>

int main(){
    using namespace std;
    string sor;
    do {
        getline(cin, sor);
        size_t hol = 0;
        while ((hol = sor.find('[', hol)) != sor.npos) {
            cout << sor.substr(hol+1, sor.find(']', hol)-hol-1) << ' ';
            hol++;
        }
    } while (sor.find('@') == sor.npos);
    return 0;
}
```

2. A feladatod, hogy olvass be két sorozat mérési eredményt (valós számok). Mindkét sorozat ugyanannyi elemből áll, de előre nem tudjuk, hogy hányból. A mérési eredményeket a szabványos bemeneten kapjuk, szóközzel elválasztva. A két sorozat adatait egy 0 érték választja el, ami nem tárolandó, minden mért érték 0-tól különböző. Pl. bemenet: 8.1 -12.2 16.5 -14.2 11.1 0 -7.3 10.4 17.5 -16.8 13.2. Az egyes részfeladatokra csak akkor jár pont, ha pontosan a következő módon oldod meg. (max. 10p)

- Olvasd be a két sorozatot az std::cin-ről két std::vector típusú tárolóba, a tárolók double típusú értékeket tároljanak!
- A két vector-t add össze std::transform algoritmus segítségével, az eredmény egy harmadik vector-ba kerüljön. Ehhez vagy az std::plus<double>() műveletet használd, vagy írd saját összeadó függvényt/funktort.
- Tetszőleges módszerrel számítsd ki az így kapott vector-ban lévő értékek átlagát, és írd ki! Ha a számítást helyesen végzed el az std::accumulate segítségével vagy iterátorral, +1 bónuszpontot kapsz.
- Oszd el az összeg vector minden elemét a c) feladatban kapott átlaggal az std::transform segítségével és saját felező függvény/funktor segítségével!
- Írd ki az így kapott vector-t!

```
#include <vector>
#include <iostream>
#include <algorithm>
#include <numeric>

double osztó = 1;

double oszt(double a) { return a / osztó; }

int main() {
    using namespace std;
    vector<double> sor1, sor2;
    double adat;
    cin >> adat;
```

```

while (adat != 0) {
    sor1.push_back(adat);
    cin >> adat;
}
sor2.resize(sor1.size());
for (size_t i = 0; i < sor2.size(); i++)
    cin >> sor2[i];
vector<double> osszeg(sor1.size());
transform(sor1.begin(), sor1.end(), sor2.begin(), osszeg.begin(), plus<double>());
osztó = accumulate(osszeg.begin(), osszeg.end(), 0.0) / osszeg.size();
cout << osztó << endl;
transform(osszeg.begin(), osszeg.end(), osszeg.begin(), oszt);
for (size_t i = 0; i < osszeg.size(); i++)
    cout << osszeg[i] << ' ';
return 0;
}

// iterátorral kiírás
for (vector<double>::iterator it = osszeg.begin(); it != osszeg.end(); it++)
    cout << *it << ' ';

```