4. labor kis ZH, 2024. május 2., csütörtök 10-12

NÉV (olvasható): NEPTUN kód:

Ejtőernyős ugrást hajtunk végre. A mozgásunk egydimenziósnak tekinthető. Természetesen van nálunk egy barometrikus magasságmérő berendezés, amely másodpercenként rögzíti aktuális magasságunk mérőszámát méterben. A programunk szabványos bemenetére küldi majd a műszer az általa mért pozíció értékeket. Előre nem tudjuk, hogy hány érték fog érkezni. A mérés végét 112m alatti érték jelzi (földetérés). Ezt már nem szabad benne hagyni az adatsorban.

Készíts programot, amely egy `std::vector`-ban tárolja a pozíció értékeket, majd elvégzi az előírt számításokat.

a) Olvasd be a mért értékeket, és tárold el (x) az ugróponttól mért távolságokat ($x_i := x_0 - x_i$), így már az ugrás kezdete lesz az origó!
b) Hozz létre egy újabb vector-t a pillanatnyi sebességek számára! Számítsd ki és tárold el a vectorban a sebességeket ($v = dx/dt$, a kiadódó m/s egységben)! Ha a deriváláshoz STL algoritmust használsz, és ezt helyesen teszed, +1 bónuszpontot kapsz.
c) Hasonló módon számítsd ki a gyorsulás értékeit is ($a = dv/dt$, $\mathrm{m/s^2}$).
d) Határozd meg, és írd ki, hogy mely időpontban nyitottuk ki az ernyőt (legnegatívabb gyorsulás). Ha STL algoritmust használsz, és ezt helyesen teszed, +1 bónuszpontot kapsz.
e) Váltsd át a v tömb m/s egységben értendő értékeit km/h-ra (szorzás 3.6-del). (Lehet új céltömbbe, de lehet a régi értékek helyén is.) Ha STL algoritmust használsz, és ezt helyesen teszed, +2 bónuszpontot kapsz.
f) Mekkora volt a legnagyobb sebességünk az esés során?

max. 10p, megoldási idő: 20 perc

```cpp
#include <numeric>

class ms2kmh{public: double operator()(double x){return 3.6*x;}}; // +1p e)-hez

int main()
{
    double xx,x0;
    std::vector<double> x;
    std::cin>>xx; x0=xx;
    while(xx>=112)
    {
        x.push_back(x0-xx);
        std::cin>>xx;
    }

    std::vector<double> v(x.size());
    std::adjacent_difference(&x[0],&x[x.size()],&v[0]); // +1p vagy
    std::adjacent_difference(x.begin(),x.end(),v.begin()); // +1p vagy
    v[0]=x[0];
    for(unsigned j=1u;j<x.size();++j)
        v[j]=x[j]-x[j-1];

    std::vector<double> a(v.size());
    std::adjacent_difference(v.begin(),v.end(),a.begin());

    double *p=std::min_element(&a[0],&a[a.size()]); // +1p vagy
    std::cout<<p-&a[0]<<'\n';

    std::vector<double>::iterator it=std::min_element(a.begin(),a.end()); // +1p vagy
    std::cout<<it-a.begin()<<'\n';

    unsigned minpos=0;
    for(unsigned i=1u;i!=a.size();++i)
        if(a[minpos]>a[i])minpos=i;
    std::cout<<minpos<<'\n';

    std::transform(v.begin(),v.end(),v.begin(),ms2kmh()); // +1p vagy
    for(unsigned j=0;j<v.size();++j)
        v[j]=3.6*v[j];

    it=std::max_element(v.begin(),v.end());
    std::cout<<*it<<'\n';

}
```