

Objektinis Programavimas

2-oji praktinė užduotis



© Remigijus Paulavičius [@RemigPau](#), Vilniaus Universitetas - 2018



Darbas su duomenimis

Turinys

- 👉 Užduoties formuluotė
- 👉 Reikalavimai versijai (v0.1)

Praktinės užduoties formuliuotė

- ☞ Parašykite programą, kuri nuskaito šiuos studentų duomenis:
 - ☞ **vardą** ir **pavardę**
 - ☞ n atliktų **namų darbų** (nd) rezultatus (10-balėje sistemoje), o taip pat galutinio **egzamino** (egz) rezultataj.
 - ☞ Tuomet iš šių duomenų, suskaičiuoja galutinį balą (galutinis):

$$\text{galutinis} = 0.4 \times \frac{\sum_{i=1}^n \text{nd}_i}{n} + 0.6 \times \text{egz}$$

Reikalavimai (v0.1) versijai (1)

- ☞ Pagal užduoties reikalavimus realizuokite programą ir atspausdinkite ekrane aktualią informaciją:
 - ☞ studento vardą ir pavardę,
 - ☞ namų darbų ir egzamino rezultatus
 - ☞ bei galutinį balą **dviejų skaičių po kablelio tikslumu**
- ☞ Papildykite programą, kad vietoj **vidurkio** būtų galima naudoti **medianą**.

Reikalavimai (v0.1) versijai (2)

- ☞ Papildykite programą taip, kad ji veiktu ir tokiu atveju, kai namų darbuų skaičius (n) yra nežinomas iš anksto, t.y. tik įvedimo metu vartotojas nusprendžia kuomet jis jau įvedė visu namų darbuų rezultatus. Šią užduotį realizuokite dviem būdais:
- ☞ naudojant C masyvus.
- ☞ naudojant `vector` ar kito tipo konteinerį.
- ☞ Papildykite, kad būtų galimybė, jog mokinio gautieji balai už namų darbus bei egzaminą būtų generuojami atsitiktinai.

Reikalavimai (v0.2) versijai (1)

Terminas: 2018-02-27

- ☞ Papildykite (v0.1) taip, kad būtų galima duomenis ne tik ivesti bet ir nuskaityti iš failų, t.y., sukurkite ir užpildykite failą **kursiokai.txt**, kurio (pleriminari) struktūra:

Pavardė	Vardas	ND1	ND2	ND3	ND4	ND5	Egzaminas
Pocius	Paulius	8	9	10	6	10	9
Makevičius	Augustinas	7	10	8	5	4	6
...							

Reikalavimai (v0.2) versijai (2)

Terminas: 2018-02-27

- ☞ Papildykite programą taip, nuskaičiuos duomenis iš failo, galutinis rezultatas (output'as) pleriminariai atrodytų taip:

Pavardė	Vardas	Gal. – vidurkis	Gal. – mediana
Makevičius	Augustinas	x . xx	y . yy
Pocius	Paulius	z . zz	q . qq
...			

Reikalavimai output'ui: studentai turi būtui surūšiuoti pagal vardus (ar pavardes) ir stulpeliai būtų gražiai "išlygiuoti".

Reikalavimai (v0.3) versijai (1)

Preliminarus terminas: 2018-03-01

- 👉 Atlikite versijos (v0.2) kodo reorganizavimą (refactoring'ą):
 - 👉 Kur tikslina, programoje naudokite (jeigu dar nenaudojote) struct'ūras
 - 👉 Funkcijas, naujus duomenų tipus (struct'ūras) perkelkite į antraštinius (**header** (*.h)) failus, t.y. tokiu būdu turėtumete projekte turėti kelis *.cpp failus, kaip ir kelis *.h failus.

Reikalavimai (v0.3) versijai (2)

Preliminarus terminas: 2018-03-01

👉 Panaudokite išimčių valdymą (**Exception Handling**):

```
try { // išimtys yra apdorojamos žemiau
      // kodas, kuris atlieka tam tikras užduotis
} catch (std::exception& e) {
      // kodas, kuris apdoroja išimtis
}
```

Kam viso to reikia? Kas atsitiks, jei failas, kuri bandote atidaryti neegzistuoja; arba bandote gauti masyvo elementą, kurio nėra?

Reikalavimai (v0.4) versijai (1)

Preliminarus terminas: 2018-03-08

- 👉 Programos veikimo greičio (spartos) analizė:
- 👉 Sūrušiuokite (padalinkite) studentus į dvi kategorijas:
 - 👉 Studentai, kurie **surinko < 60%** už namų darbų užduotis ("vargšiukai", "nuskriaustukai" ir pan.)
 - 👉 Studentai, kurie **surinko >= 60%** ("kietiakai", "galvočiai", "vizunčikai" ir pan.) ir buvo prileisti prie egzamino.

Versijos (v0.1) kūrimas (1)

Skaičių po kablelio tikslumo valdymas

- 👉 <iomanip> antraštės (header) faile deklaruotas manipulatorius setprecision, kuris leidžia kontroliuoti kiek reikšmingų skaitmenų išvedima informacija naudotų.
- 👉 Kai naudojame endl, kuris taip pat yra manipulatorius, mums nereikia ištraukti <iomanip> antraštės. **Kodėl?**

Versijos (v0.1) kūrimas (2)

std::setprecision pavyzdys¹

```
#include <iostream>
#include <iomanip>
#include <cmath>
#include <limits>
int main() {
    const long double pi = std::acos(-1.L);
    std::cout << "default precision (6): " << pi << '\n'
        << "std::setprecision(10): " << std::setprecision(10) << pi << '\n'
        << "max precision: "
        << std::setprecision(std::numeric_limits<long double>::digits10 + 1) << pi << '\n';
}
```

default precision (6): 3.14159

std::setprecision(10): 3.141592654

max precision: 3.141592653589793239

¹ <http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2013/n3690.pdf>

Versijos (v0.1) kūrimas (3)

Rezultatų spausdinimas 2 skaičių po kablelio tikslumu

```
// išsaugome numatytaį (default) tikslumą
streamszie prec = cout.precision(); // C++11: auto prec
cout << "Galutinis balas yra " << setprecision(3)
      << 0.6 * egzaminas + 0.4 * suma / n
      << setprecision(prec) << endl;
```

Versijos (v0.1) kūrimas (4)

EOF (end-of-file) signalas

- ☞ Vienas iš būdų, kaip užbaigti duomenų įvedimą (cikle) yra **end-of-file (EOF)** signalo pasiuntimas.
- ☞ Skirtingos C++ realizacijos siūlo skirtinges būdus kaip tokį signalą į programą pasiūsti. Labiausiai paplitęs būdas yra:
 - ☞ pradėti naują eilutę (spaudžiame Enter)
 - ☞ tuomet spaudžiame Ctrl+z (Windows) arba Ctrl+d (Unix).

Versijos (v0.1) kūrimas (5)

EOF (end-of-file) pavyzdžiai²

```
while( !cin.eof() ) { // EOF false čia
    cin >> x;          // kai nuskaityti nepavyksta, EOF tampa true
    // use x           // naudojame x, nors nieko nenuskaitėme
}
```

VS.

```
while(cin >> x) { // Bando nuskaityti į x, grąžina false kai nepavyksta
                    // ciklą vykdys tik kai nuskaityti pavyko
}
```

² <https://stackoverflow.com/questions/4533063/how-does-ifstreams-eof-work>

Versijos (v0.1) kūrimas (6)

while (cin >> x) reiškinio analizė

Norint nuskaityti reikšmę į **x** ir patikrinti ar pavyko:

```
if (cin >> x) { /*...*/ }
```

Tai ekvivalentu:

```
cin >> x;  
if (cin) { /* ... */ }
```

Todėl cin naudojimas **if** ar **while** sąlygose ekvivalentus: ar paskutinis bandymas nuskaityti iš `cin buvo sėkmingas.

Versijos (v0.1) kūrimas (7)

Kada skaitymas iš srauto (cin) gali būti nesėkminges?

- 👉 Mes galėjome pasiekti įvesties failo pabaigą.
- 👉 Galbūt susidūrēme su įvestimi, nesuderinama su kintamojo tipu, kurių bandome perskaityti, pvz., taip gali atsitikti, jei mes bandome perskaityti int bet reikšmę yra nėra skaičius.
- 👉 Sistemoje gali būti aptikta įvesties įrangos gedimas.

Visais šiai atvejais: naudojant sąlygoje cin reikšmę bus false.

Versijos (v0.1) kūrimas (8)

Srauto (stream) būsenos (1)

Srautas turi būseną (konstantą), kuri nusako, ar I/O (**Input**/
Output) buvo sėkminga, o jei ne - kokia nesėkmės priežastis.

Būsena	Reikšmė
goodbit	Viskas OK
eofbit	Pasiekta failo pabaiga
failbit	Klaida; I/O operacija nebuvo sėkminga
badbit	Esminė klaida; neapibrėžta būsena

Versijos (v0.1) kūrimas (9)

Srauto (stream) būsenos (2)

- 👉 **failbit** - reiškia, kad operacija nebuvo tinkamai apdorota, bet srautas yra OK. Pavyzdžiui, tai nutinka, jei skaitmuo turėjo būti nuskaitomas, bet gautas simbolis yra raidė.
- 👉 **badbit** - reiškia, kad srautas yra sugadintas arba ryšys su srautu yra prarastas.
- 👉 **eofbit** - tradiciškai nutinka kartu su **failbit**, nes failo pabaiga (`end-of-file`) nustato **eofbit**, bet tuo pačiu ir **failbit**, nes nieko nuskaityti nepavyko.

Versijos (v0.1) kūrimas (10)

Nežinomo skaičiaus namų darbų (nd) nuskaitymas

```
int n = 0;           // nd skaitiklis
double suma = 0.;   // nd įverčiu suma
double x;           // kintamasis i kuri nuskaityti
/* invariantas: iki šiol nuskaitėme `n' nd rezultatu,
   ir jų įverčiu suma lygi `suma' */
while (cin >> x) {
    ++n;             // reikalauja pirmoji dalis invarianto
    suma += x;        // reikalauja antroji dalis invarianto
}
```

Versijos (v0.1) kūrimas (11)

Medianos skaičiavimas

- 👉 Norėdami apskaičiuoti medianą, privalome:
- 👉 Išsaugoti nuskaitytas reikšmes, nežinant iš anksto, kiek reikšmių bus.
- 👉 Kai visos reikšmės nuskaitytos - išrūšiuoti jas.
- 👉 Gauti vidutinę(-es) reišmę(-es) efektyviai.

Versijos (v0.1) kūrimas (12)

Nežinomo skaičiaus namų darbų (nd) nuskaitymas į vektorių

```
// Aukščiau: using std::vector
vector<double> nd; // vektorius su namų darbų rezultatais
double x; // kintamasis į kurį nuskaityti
// invariantas: `nd' kaupia visus iki šiol nuskaitytus ND
while (cin >> x) {
    nd.push_back(x); // Pridedame nuskaitytą rezultatą
}
```

Versijos (v0.1) kūrimas (13)

typedef ir using keywords'ai

“A **typedef**-name can also be introduced by an alias-declaration. The identifier following the **using** keyword becomes a **typedef**-name and the optional attribute-specifier-seq following the identifier appertains to that **typedef**-name. It has the same semantics as if it were introduced by the **typedef** specifier. In particular, it does not define a new type and it shall not appear in the type-id.¹”

¹ <http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2013/n3690.pdf>

Versijos (v0.1) kūrimas (14)

typedef ir using pavyzdžiai

```
typedef int MyInt;  
// ekvivalentu:  
using MyInt = int; // Nuo C++11  
// Tuomet:  
MyInt x = 5;  
  
// Patogu, kai ilgi tipai:  
using MyIntVector = std::vector<int>;  
// Ekvivalentu:  
typedef std::vector<int> MyIntVector;  
// Tuomet:  
MyIntVector iVect(10, 1);
```

Versijos (v0.1) kūrimas (15)

```
// patikriname ar įvedė nd rezultatus
typedef vector<double>::size_type vecSize;
vecSize size = nd.size();
if (size == 0) { // `ekvivalentu': if (nd.empty())
    cout << "Privalote įvesti ND rezultatus."
          << "Bandykite iš naujo. \n";
    return 1;
}
```

👉 Kodėl `return 1;` ?

Versijos (v0.1) kūrimas (16)

```
// surušiuojame ND, naudojant `sort` iš <algorithm>
sort(nd.begin(), nd.end());
// apskaičiuojame medianą
vecSize vid = size/2; // vidurinis elementas
double mediana;
mediana = size % 2 == 0 ? (nd[vid-1] + nd[vid]) / 2
                         : nd[vid];
```

- 👉 Sąlyginis operatorius yra santrumpa **if-then-else** išraiškai ir dažnai vadinamas ? : operatoriumi.

Klausimai!

!?

