Objektinis programavimas

Praktinės užduotys

dr. Remigijus Paulavičius @RemigPau

Vilniaus Universitetas

1. Įrėmintas pasisveikinimas

1. Įrėmintas pasisveikinimas - modernesnis “Sveikas Pasauli!”

Užduotis 1: Suprogramuokite modernesnę “Sveikas Pasauli!” versiją taip, kad programos įvedimas ir išvedimas atrodytų taip:

```
Įveskite savo vardą: Remigijus

**********************
*                    *
* Sveikas, Remigijus! *
*                    *
**********************
```

Reikalavimai programai:

1. Programa turi atspausdinti penkias eilutes:
   - Pirmoje eilutėje prasideda "rėmelis", kuris yra seka * simbolių, ir kurio plotis priklauso nuo žmogaus vardo (įvesto programos vartotojo), pasisveikinimo "Sveikas, ", tarpo ir * simbolių pradžioje ir pabaigoje.
   - Antroji eilutė prasideda ir baigiasi * simboliu, o vidus užpildytas reikiamu skaičiumi tarpo simbolių.
   - Trečioji eilutė susideda iš *, tarpo, pasisveikinimo, tarpo ir vėl *.
   - Kervirta ir penkta eilutės bus analogiškos antrąjai ir pirmąjai.
2. Modifikuokite programą taip, kad kai vartotojas yra moteris, tuomet vietoj "**Sveikas,** " rašytų "**Sveika,** ".
3. Modifikuokite programą taip, kad nereiktų kiekvienos eilutės saugoti atskirame kintamajame (vos atspausdinus, jie tampa nebereikalingi), t.y., tik pasisveikinimo tekstas tūri būti saugomas jam skirtame kintamajame.
4. Kur yra tikslinga, naudokite C++11 standarte atsiradusius patobulinimus.
5. Nusiųsti (angl. push'inti) programą į savo asmeninį profilį vienoje iš versijų kontroliavimo platformų: Github, Bitbucket, GitLab ir pan.

2. Duomenų apdorojimas

2. Duomenų apdorojimas (1)

Užduotis 2
Parašykite programą, kurioje yra nuskaitomi įvedami tokie duomenys:

- mokinio/studento **vardas ir pavardė**
- jo n atliktų **namų darbų** ($nDarbas_i$) surinkti balai (10-balėje sistemoje)
- galutinio **egzamino** rezultatas (10-balėje sistemoje)

Pagal įvestus duomenis apskaičiuokite **galutinį balą** (*galBalas*) pagal tokią formulę:

$$galBalas = 0.4 \times \frac{\sum_{i=1}^{n} nDarbas_i}{n} + 0.6 \times egzaminas$$

Reikalavimai programos versijai v0.1:

v0.1 Realizuokite pirminę programos versiją pagal užduoties aprašymą ir atspausdinkite ekrane visą aktualią informaciją: **studento vardą ir pavardę, jo namų darbų ir egzamino rezultatus bei galutinį balą (dviejų skaičių po kablelio tikslumu)**

1. Papildykite programą, kad vietoj namų darbų **vidurkio įverčio** $\left(\sum_{i=1}^{n} nDarbas_i/n\right)$ būtų galima naudoti **medianą**.
   Kaip manote kuris įvertis yra sąžiningesnis?
2. Papildykite programą taip, kad ji veiktų ir tokiu atveju, kai namų darbų skaičius (n) yra nežinomas iš anksto, t.y. tik įvedimo metu vartotojas *nusprendžia* kuomet jis jau įvedė visų namų darbų rezultatus.
   Šią užduotį (papildymą) realizuoti reiktų dviem būdais:
   a) **C būdu**: naudojant masyvus.
   b) **C++ būdu**: naudojant `vector` ar kito tipo konteinerį.
3. Papildykite programą taip, kad būtų galimybė, jog mokinio gautieji balai už namų darbus bei egzaminą būtų generuojami atsitiktinai.

2. Duomenų apdorojimas (2)

Reikalavimai programos versijai v0.2:

v0.2 Papildykite (v0.1) taip, kad būtų galima duomenis ne tik įvesti bet ir **nuskaityti iš failų**.

4. 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
...
```

5. Papildykite programą taip, nuskaičiuos duomenis iš failo, galutinis rezultatas (**output**'as) pleriminariai atrodytų taip:

```
Pavardė      Vardas      Galutinis-vidurkis   Galutinis-mediana
Makevičius   Augustinas  x.xx                 y.yy
Pocius       Paulius     z.zz                 q.qq
...
```

Reikalavimai output'ui: Kad studentai būtų surūšiuoti pagal pavardes ir stulpeliai būtų gražiai "išlygiuoti".

2. Duomenų apdorojimas (3)

Reikalavimai programos versijai v0.3:

v0.3 Reorganizuokite (atlikite **code refactoring**) versijos (v0.2) realizaciją:

- Kur tikslinga, programoje susikurkite ir naudokite `struct`'ūras
- Kur tikslinga, panaudokite išimčių valdymą (angl. **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? O pvz. kas atsitiks, jeigu failas, kurį bandote atidaryti neegzistuoja; arba bandote gauti masyvo elementą (indeksą), kuris neegzistuoja?
- Funkcijas, naujus duomenų tipus (struct'ūras) perkelkite į antraštinius (angl. **header**) failus, t.y. tokiu būdu turėtumete turėti ne vieną, o kelis `*.cpp` failus, kaip ir `*.h` failus.