KAUNO TECHNOLOGIJOS UNIVERSITETAS

Objektinis programavimas (P175B502)

Laboratorinio darbo ataskaita

Atliko:

IF-1/9 gr. studentas

Tautvydas Petkus

Priėmė:

Prof. Vacius Jusas

KAUNAS 2012

TURINYS

1. Dinaminis objektų masyvas (L1) 3

1.1. Darbo užduotis 3

1.2. Grafinės vartotojo sąsajos schema 3

1.3. Sąsajoje panaudotų komponentų keičiamos savybės 3

1.4. Programos vartotojo vadovas 3

1.5. Programos tekstas 3

1.6. Pradiniai duomenys ir rezultatai 3

2. Vienkryptis sąrašas (L2) 3

2.1. Darbo užduotis 3

2.2. Grafinės vartotojo sąsajos schema 3

2.3. Sąsajoje panaudotų komponentų keičiamos savybės 3

2.4. Programos vartotojo vadovas 3

2.5. Programos tekstas 3

2.6. Pradiniai duomenys ir rezultatai 3

3. Dvikryptis sąrašas (L3) 3

3.1. Darbo užduotis 3

3.2. Grafinės vartotojo sąsajos schema 3

3.3. Sąsajoje panaudotų komponentų keičiamos savybės 3

3.4. Programos vartotojo vadovas 3

3.5. Programos tekstas 3

3.6. Pradiniai duomenys ir rezultatai 3

4. Objektų sąrašas (L4) 3

4.1. Darbo užduotis 3

4.2. Grafinės vartotojo sąsajos schema 3

4.3. Sąsajoje panaudotų komponentų keičiamos savybės 3

4.4. Programos vartotojo vadovas 3

4.5. Programos tekstas 3

4.6. Pradiniai duomenys ir rezultatai 3

5. Šakotas sąrašas (L5) 3

5.1. Darbo užduotis 3

5.2. Grafinės vartotojo sąsajos schema 3

5.3. Sąsajoje panaudotų komponentų keičiamos savybės 3

5.4. Programos vartotojo vadovas 3

5.5. Programos tekstas 3

5.6. Pradiniai duomenys ir rezultatai 3

# Dinaminis objektų masyvas (L1)

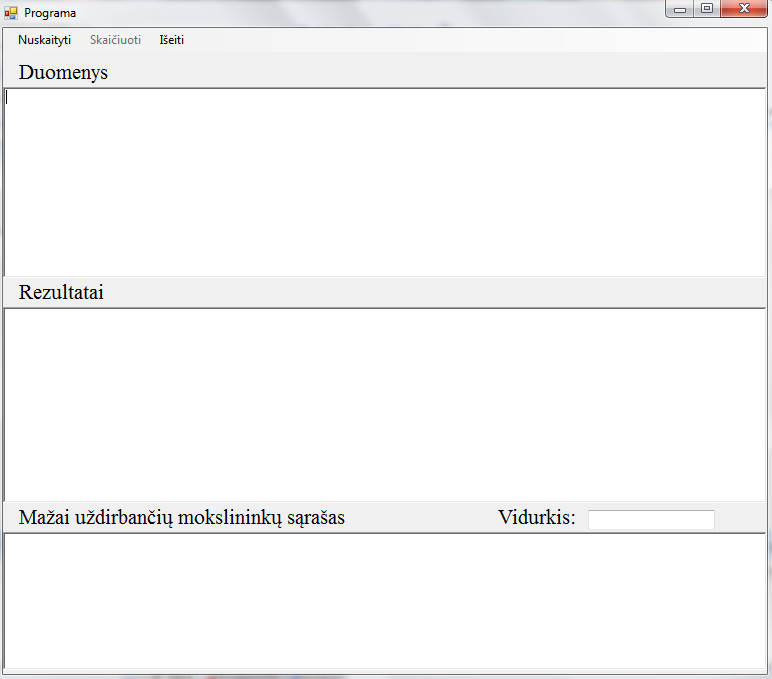
## Darbo užduotis

39. Moksliniai darbuotojai atliko darbus 4 skirtingose temose. Visos temos gavo premijas. Duomenų faile tokia informacija:

Pirmoje eilutėje – premijų dydžiai.

Tolesnėse eilutėse – darbuotojų indėliai, kurie išreikšti naudingumo koeficientu, į eilinę temą. Informacija apie darbuotoją užima vieną eilutę. Remiantis darbuotojų indėliais, suskaičiuoti kiekvienam darbuotojui priklausančios premijos dydį pagal kiekvieną temą atskirai ir bendrą premijų sumą. Suformuoti sąrašą darbuotojų, kurie uždirbo mažiau už vidurkį. Rikiuoti(pavardė, vardas).

## Grafinės vartotojo sąsajos schema



## Sąsajoje panaudotų komponentų keičiamos savybės

|  |  |  |
| --- | --- | --- |
| Komponentas | Savybė | Reikšmė |
| Meniu juosta  (ToolStripMenu) | Rodo, kokias funkcijas atlieka programa | Ja pasinaudojus galima nuskaityti duomenis, juos apsikačiuoti, baigti darbą su programa |
| Tekstinių failų laukeliai (richtextbox1, richtextbox2, richtextbox3) | Pavaizduoja atitinkamų tekstinių failų turinį | Parodo nuskaitytų pradinių duomenų failą, rezultatų failą su skaičiavimais ir rikiavimu. |
| Vidurkio reikšmės laukelis (textbox1) | Pavaizduoja, koks buvo vidurkis, pagal kurį buvo atrinktas mažai uždirbančių mokslininkų sąrašas | Padeda vartotojui lengviau įvertinti duomenis |
| Laukelių pavadinimai (label1, label2, label3, label4) | Skirti pavaizduoti antraštėms ar tekstui programos lauke | Skirta vartotojui atskirti duomenų, rezultatų failą, taip pat atskirti atrinktą rezultatų failą bei nurodant textbox1 reikšmę programai. |

## Programos vartotojo vadovas

1. Visi duomenys turi būti surašyti tekstiniame faile Duomenys.txt, pagal užduoties sąlygoje nurodytus reikalavimus.
2. Spauskite mygtuką „Nuskaityti“, kad nuskaitytumėte duomenis iš failo.
3. Atsilaisvinus mygtukui „Skaičiuoti“, spauskite jį, norėdami apskaičiuoti kiekvieno mokslininko uždirbtą premiją bei visas premijas kartu.

Tuo pačiu bus apskaičiuotas vidurkis visų mokslininkų uždirbtų pinigų, kuris bus pavaizduotas laukelyje „Vidurkis“.

3-iajame teksto lauke bus pavaizduotas sąrašas mokslininkų, kurių premijų suma nesiekė vidurkio.

1. Norėdami baigti darbą, spauskite „Išeiti“.

## Programos tekstas

**Mokslininkai.h:**

#pragma once

#include <string>

#include <iostream>

#include <fstream>

#include <sstream>

#include <iomanip>

using namespace std;

class Mokslininkai

{

public:

static const int Cpremijos=4;

private:

string kodas, vardas, pavarde;

double ind[Cpremijos], pin[Cpremijos] , S;

public:

Mokslininkai() {}

Mokslininkai(string kodasR, string vardasR, string pavardeR, double indR[]);

string Spausdinti();

string SpausdintiPapildomai();

void ApskaiciuotiS();

void DetiUzdirbius(double a, int b);

double ImtiS();

double ImtiIndelius(int i);

bool operator > (Mokslininkai & C);

};

**Mokslininkai.cpp:**

#include "StdAfx.h"

#include "Mokslininkai.h"

using namespace std;

Mokslininkai::Mokslininkai(string kodasR, string vardasR, string pavardeR, double indR[]) :

kodas(kodasR), vardas(vardasR), pavarde(pavardeR), S(0)

{

for (int i = 0; i < Cpremijos; i++){

ind[i] = indR[i];

pin[i] = 0;

}

}

string Mokslininkai::Spausdinti()

{

stringstream ss;

ss << left << setw(15) << kodas << setw(15) << vardas << setw(15) << pavarde;

for (int i = 0; i < Cpremijos; i++)

ss << right << setw(4) << ind[i];

return ss.str();

}

string Mokslininkai::SpausdintiPapildomai()

{

stringstream ss;

ss << left << setw(15) << kodas << setw(15) << vardas << setw(15) << pavarde;

for (int i = 0; i < Cpremijos; i++)

ss << right << setw(4) << ind[i];

for (int i = 0; i < Cpremijos; i++)

ss <<' '<< right << setw(6) << ((int)pin[i]);

ss << right << setw(10) << ((int)S);

return ss.str();

}

//----------------------------------

double Mokslininkai::ImtiS() { return S; }

//----------------------------------

double Mokslininkai::ImtiIndelius(int i)

{

return ind[i];

}

//----------------------------------

void Mokslininkai::ApskaiciuotiS(){

for (int i = 0; i < Cpremijos; i++)

S+= pin[i];

}

//----------------------------------

void Mokslininkai::DetiUzdirbius(double a, int b)

{

pin[b] = a;

}

//------------------------------------

bool Mokslininkai::operator > (Mokslininkai & C)

{

return pavarde > C.pavarde || ((pavarde == C.pavarde) && (vardas > C.vardas));

}

**Fondas.h:**

#pragma once

#include "Mokslininkai.h"

#include <string>

#include <iostream>

#include <fstream>

#include <sstream>

#include <iomanip>

using namespace std;

class Fondas

{

public:

static const int Cpremijos=4;

static const int Cmoksl=32;

private:

double Premija[Cpremijos];

Mokslininkai \*M;

int n;

public:

Fondas();

~Fondas();

void DetiKiek(int a);

int Kiek();

Mokslininkai Imti(int i);

void DetiP(double p, int i);

void Deti(Mokslininkai p);

double ImtiP (int i);

void Paskirstymas();

void Rikiavimas();

};

**Fondas.cpp:**

#include "StdAfx.h"

#include "Fondas.h"

using namespace std;

Fondas::Fondas() : n(0), M(NULL)

{

M = new Mokslininkai[Cmoksl];

}

Fondas::~Fondas()

{

delete [] M;

}

void Fondas::DetiKiek(int a) { n = a; }

//--------------------------------------------------------

int Fondas::Kiek()

{ return n; }

//--------------------------------------------------------

Mokslininkai Fondas::Imti(int i)

{ return M[i];}

//--------------------------------------------------------

void Fondas::Deti(Mokslininkai p) {

if (n < Cmoksl)

M[n] = p;

}

//--------------------------------------------------------

void Fondas::DetiP(double p, int i)

{

Premija[i] = p;

}

//--------------------------------------------------------

double Fondas::ImtiP(int i)

{

return Premija[i];

}

//--------------------------------------------------------

void Fondas::Paskirstymas() {

double indS = 0;

double pinigai;

for (int i = 0; i < Cpremijos; i++){

indS = 0;

for (int j = 0; j < n; j++)

indS+= M[j].ImtiIndelius(i);

for (int j = 0; j < n; j++){

pinigai = (M[j].ImtiIndelius(i) / indS) \* Premija[i];

M[j].DetiUzdirbius(pinigai, i);

}

}

for (int i = 0; i < n; i++)

M[i].ApskaiciuotiS();

}

//-------------------------------------------------------

void Fondas::Rikiavimas()

{

Mokslininkai CC;

for (int i = 0; i < n - 1; i++)

for (int j = i + 1; j < n; j++)

if (M[i] > M[j])

{

CC = M[i];

M[i] = M[j];

M[j] = CC;

}

}

**Form1.h:**

#pragma once

#include <string>

#include <iostream>

#include <fstream>

#include <sstream>

#include <iomanip>

#include "Mokslininkai.h"

#include "Fondas.h"

const string Cduom = "Duomenys.txt";

const string Crez = "Rezultatai.txt";

const string Crez2= "Rezultatai2.txt";

const string Crez3= "Rezultatai3.txt";

const double Cprem = 4;

namespace S2L1 {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

void Duomenys(Fondas \* F)

{

ifstream duom(Cduom.c\_str());

string kod, pav, vard;

double indelis[Mokslininkai::Cpremijos];

double p[Fondas::Cpremijos];

F->DetiKiek(0);

for (int i = 0; i < Fondas::Cpremijos; i++)

duom >> p[i];

for (int i = 0; i < Fondas::Cpremijos; i++)

F->DetiP(p[i], i);

while(!duom.eof()) {

getline (duom, kod, ',');

getline (duom, pav, ',');

getline (duom, vard, ',');

for (int i=0; i<Mokslininkai::Cpremijos; i++)

duom >> indelis[i];

F->Deti(Mokslininkai(kod, pav, vard, indelis));

F->DetiKiek(F->Kiek() + 1);

}

duom.close();

}

//-----------------------------------------------------

// Spausdinami objekto F duomenys

void Isvesti(Fondas \* F)

{

ofstream rez(Crez.c\_str(), ios::app);

for (int i = 0; i < Fondas::Cpremijos; i++)

rez <<' '<< F->ImtiP(i);

rez << endl << left << setw(15) << "Kodas" << setw(15) << "Pavarde" << setw(15) <<"Vardas"<< setw(15) << "Indeliai \n";

rez << "-----------------------------------------\n";

for (int i = 0; i < F->Kiek(); i++)

rez << F->Imti(i).Spausdinti();

rez << endl << "--------------------------------------\n\n\n";

rez.close();

}

//-----------------------------------------------------

void Apskaiciavimas(Fondas \* F, Fondas \* R)

{

R->DetiKiek(0);

for (int i = 0; i < Fondas::Cpremijos; i++)

R->DetiP(F->ImtiP(i), i);

for (int i = 0; i < F->Kiek(); i++){

R->Deti(F->Imti(i));

R->DetiKiek(R->Kiek() + 1);

}

R->Paskirstymas();

R->Rikiavimas();

}

//-----------------------------------------------------

void RezultatuIsvedimas(Fondas \* R, string rezFailas)

{

ofstream rez(rezFailas.c\_str(), ios::app);

rez << endl << left << setw(15) << "Kodas" << setw(15) << "Pavarde" << setw(15) <<"Vardas"<< setw(20) << "Indeliai" <<

setw(30) << "Premijos" << setw(10) << "Suma" << endl;

rez << "-----------------------------------------------------------------------------------------\n";

for (int i = 0; i < R->Kiek(); i++)

rez << R->Imti(i).SpausdintiPapildomai();

rez << endl;

rez << "-----------------------------------------------------------------------------------------\n";

rez.close();

}

//----------------------------------------------------

double Vidurkiai(Fondas \* R, Fondas \* U)

{

double PremijuV = 0;

U->DetiKiek(0);

for (int i = 0; i < R->Kiek(); i++)

PremijuV += R->Imti(i).ImtiS();

PremijuV = PremijuV / R->Kiek();

for (int i = 0; i < R->Kiek(); i++)

if (R->Imti(i).ImtiS() < PremijuV)

{

U->Deti(R->Imti(i));

U->DetiKiek(U->Kiek() + 1);

}

return PremijuV;

}

//-----------------------------------------------------

/// <summary>

/// Summary for Form1

///

/// WARNING: If you change the name of this class, you will need to change the

/// 'Resource File Name' property for the managed resource compiler tool

/// associated with all .resx files this class depends on. Otherwise,

/// the designers will not be able to interact properly with localized

/// resources associated with this form.

/// </summary>

public ref class Form1 : public System::Windows::Forms::Form

{

public:

Form1(void)

{

InitializeComponent();

//

//TODO: Add the constructor code here

//

info = new Fondas;

infoc = new Fondas;

infoc2 = new Fondas;

}

protected:

/// <summary>

/// Clean up any resources being used.

/// </summary>

~Form1()

{

if (components)

{

delete components;

}

}

private: Fondas \*info, \*infoc, \*infoc2;

private: System::Windows::Forms::MenuStrip^ menuStrip1;

protected:

private: System::Windows::Forms::ToolStripMenuItem^ nuskaitytiToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ išeitiToolStripMenuItem;

private: System::Windows::Forms::RichTextBox^ richTextBox1;

private: System::Windows::Forms::Label^ label1;

private: System::Windows::Forms::Label^ label2;

private: System::Windows::Forms::RichTextBox^ richTextBox2;

private: System::Windows::Forms::Label^ label3;

private: System::Windows::Forms::RichTextBox^ richTextBox3;

private: System::Windows::Forms::ToolStripMenuItem^ skaičiuotiToolStripMenuItem;

private: System::Windows::Forms::Label^ label4;

private: System::Windows::Forms::TextBox^ textBox1;

private:

/// <summary>

/// Required designer variable.

/// </summary>

System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code {.....}

#pragma endregion

private: System::Void Form1\_Load(System::Object^ sender, System::EventArgs^ e) {

}

private: System::Void išeitiToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) {

Close();

}

private: System::Void nuskaitytiToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) {

ofstream rez(Crez.c\_str());

rez.close();

ofstream rez2(Crez2.c\_str());

rez2.close();

ofstream rez3(Crez3.c\_str());

rez3.close();

Duomenys(info);

Isvesti(info);

richTextBox1->LoadFile("Rezultatai.txt", RichTextBoxStreamType::PlainText);

skaičiuotiToolStripMenuItem->Enabled = true;

}

private: System::Void richTextBox1\_TextChanged(System::Object^ sender, System::EventArgs^ e) {

}

private: System::Void menuStrip1\_ItemClicked(System::Object^ sender, System::Windows::Forms::ToolStripItemClickedEventArgs^ e) {

}

private: System::Void richTextBox2\_TextChanged(System::Object^ sender, System::EventArgs^ e) {

}

private: System::Void skaičiuotiToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) {

Apskaiciavimas(info, infoc);

double vidurkis = Vidurkiai(infoc, infoc2);

RezultatuIsvedimas(infoc, Crez2);

RezultatuIsvedimas(infoc2, Crez3);

richTextBox2->LoadFile("Rezultatai2.txt", RichTextBoxStreamType::PlainText);

richTextBox3->LoadFile("Rezultatai3.txt", RichTextBoxStreamType::PlainText);

textBox1->Text = ((int)vidurkis).ToString();

}

private: System::Void textBox1\_TextChanged(System::Object^ sender, System::EventArgs^ e) {

}

};

}

## Pradiniai duomenys ir rezultatai

**Duomenys.txt:**

5000 2500 8000 1000

38604163754, Hhhhhh, Hhhhhhhhh, 5 9 7 2

37407011234, Gggggg, Ggggggggg, 2 1 0 3

39012128243, Ffffff, Fffffffff, 6 7 1 9

36401288430, Eeeeee, Eeeeeeeee, 1 2 3 4

37702131348, Dddddd, Ddddddddd, 9 4 3 6

37103044686, Cccccc, Ccccccccc, 4 5 4 6

33008141568, Aaaaaa, Aaaaaaaaa, 7 2 2 3

39011128434, Bbbbbb, Bbbbbbbbb, 7 6 1 9

**Rezultatai.txt:**

5000 2500 8000 1000

Kodas Pavarde Vardas Indeliai

-----------------------------------------

38604163754 Hhhhhh Hhhhhhhhh 5 9 7 2

37407011234 Gggggg Ggggggggg 2 1 0 3

39012128243 Ffffff Fffffffff 6 7 1 9

36401288430 Eeeeee Eeeeeeeee 1 2 3 4

37702131348 Dddddd Ddddddddd 9 4 3 6

37103044686 Cccccc Ccccccccc 4 5 4 6

33008141568 Aaaaaa Aaaaaaaaa 7 2 2 3

39011128434 Bbbbbb Bbbbbbbbb 7 6 1 9

--------------------------------------

**Rezultatai2.txt:**

Kodas Pavarde Vardas Indeliai Premijos Suma

-----------------------------------------------------------------------------------------

33008141568 Aaaaaa Aaaaaaaaa 7 2 2 3 853 138 761 71 1825

39011128434 Bbbbbb Bbbbbbbbb 7 6 1 9 853 416 380 214 1865

37103044686 Cccccc Ccccccccc 4 5 4 6 487 347 1523 142 2501

37702131348 Dddddd Ddddddddd 9 4 3 6 1097 277 1142 142 2661

36401288430 Eeeeee Eeeeeeeee 1 2 3 4 121 138 1142 95 1498

39012128243 Ffffff Fffffffff 6 7 1 9 731 486 380 214 1813

37407011234 Gggggg Ggggggggg 2 1 0 3 243 69 0 71 384

38604163754 Hhhhhh Hhhhhhhhh 5 9 7 2 609 625 2666 47 3949

-----------------------------------------------------------------------------------------

**Rezultatai3.txt:**

**Kodas Pavarde Vardas Indeliai Premijos Suma**

**-----------------------------------------------------------------------------------------**

**33008141568 Aaaaaa Aaaaaaaaa 7 2 2 3 853 138 761 71 1825**

**39011128434 Bbbbbb Bbbbbbbbb 7 6 1 9 853 416 380 214 1865**

**36401288430 Eeeeee Eeeeeeeee 1 2 3 4 121 138 1142 95 1498**

**39012128243 Ffffff Fffffffff 6 7 1 9 731 486 380 214 1813**

**37407011234 Gggggg Ggggggggg 2 1 0 3 243 69 0 71 384**

**-----------------------------------------------------------------------------------------**

# Vienkryptis sąrašas (L2)

## Darbo užduotis

## Grafinės vartotojo sąsajos schema

## Sąsajoje panaudotų komponentų keičiamos savybės

|  |  |  |
| --- | --- | --- |
| Komponentas | Savybė | Reikšmė |
|  |  |  |
|  |  |  |
|  |  |  |

## Programos vartotojo vadovas

## Programos tekstas

## Pradiniai duomenys ir rezultatai

# Dvikryptis sąrašas (L3)

## Darbo užduotis

## Grafinės vartotojo sąsajos schema

## Sąsajoje panaudotų komponentų keičiamos savybės

|  |  |  |
| --- | --- | --- |
| Komponentas | Savybė | Reikšmė |
|  |  |  |
|  |  |  |
|  |  |  |

## Programos vartotojo vadovas

## Programos tekstas

## Pradiniai duomenys ir rezultatai

# Objektų sąrašas (L4)

## Darbo užduotis

## Grafinės vartotojo sąsajos schema

## Sąsajoje panaudotų komponentų keičiamos savybės

|  |  |  |
| --- | --- | --- |
| Komponentas | Savybė | Reikšmė |
|  |  |  |
|  |  |  |
|  |  |  |

## Programos vartotojo vadovas

## Programos tekstas

## Pradiniai duomenys ir rezultatai

# Šakotas sąrašas (L5)

## Darbo užduotis

## Grafinės vartotojo sąsajos schema

## Sąsajoje panaudotų komponentų keičiamos savybės

|  |  |  |
| --- | --- | --- |
| Komponentas | Savybė | Reikšmė |
|  |  |  |
|  |  |  |
|  |  |  |

## Programos vartotojo vadovas

## Programos tekstas

## Pradiniai duomenys ir rezultatai