МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Вятский государственный университет»
Факультет автоматики и вычислительной техники
Кафедра электронных вычислительных машин

Лабораторная работа № 6 по курсу «Технологии программирования»

Выполнил студент группы ИВТ-21 _____/Рзаев А. Э./
Проверил доцент кафедры ЭВМ _____/Долженкова М. Л./

Киров 2017

1 Задание

Выбрать предметную область, в которой разработать иерархию классов. Класс должен иметь собственные член-данные и член-функции. Дочерний класс должен иметь наследуемые и перекрывающие член-функций. Разработать приложение, демонстрирующее принципы наследования, инкапсуляции и полиморфизма.

2 Результат работы

Экранные формы приведены в приложении А.

3 Иерархия классов

Иерархия классов приведена в приложении Б.

4 Листинг программы

Листинг программы приведен в приложении В.

5 Вывод

В ходе выполнения лабораторной работы была написана программа с ГПИ на wxWidgets для расчета сопротивления проводников двух видов: металлов и растворов солей. Для реализации необходимого функционала были разработаны три класса: один базовый и два дочерних класса. Базовый класс имеет методы для чтения и записи удельного сопротивления и длины проводника, расчета сопротивления и проверки введенных данных. Дочерний классы определяют свои дополнительные методы для изменения специфичных параметров проводников, а также переопределяют методы для расчета сопротивления и проверки введенных данных.

Приложение А
(обязательное)
Экранные формы

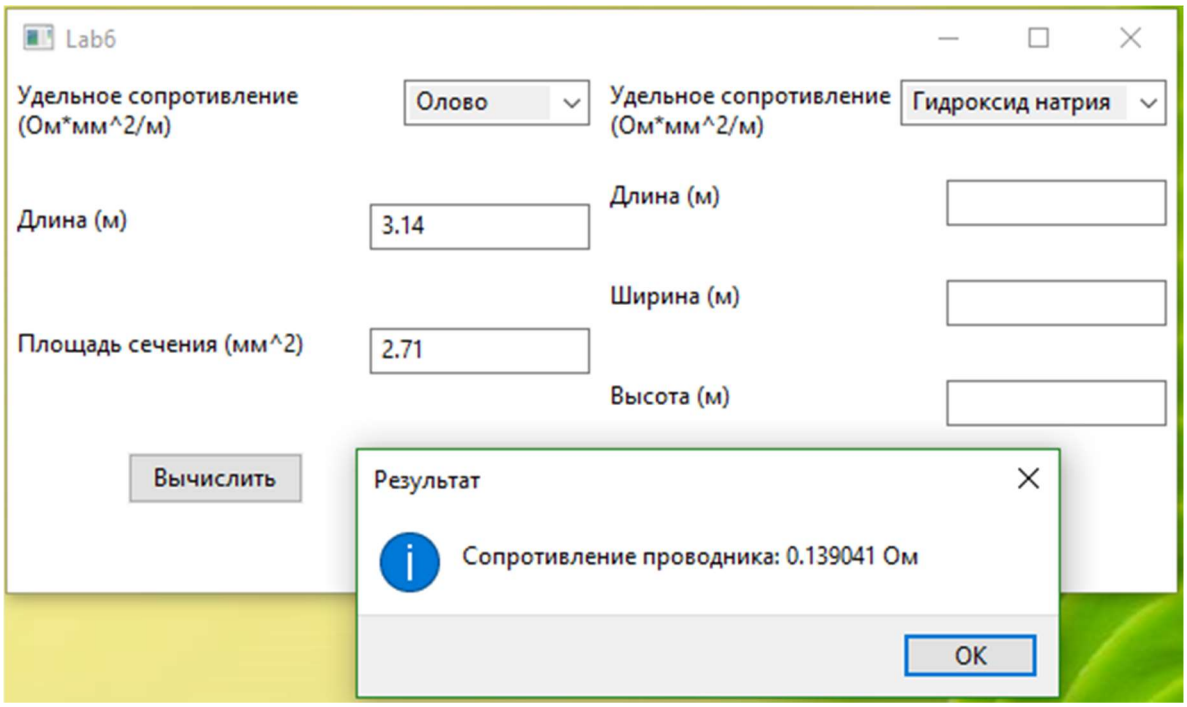


Рисунок 1 – Результат работы программы

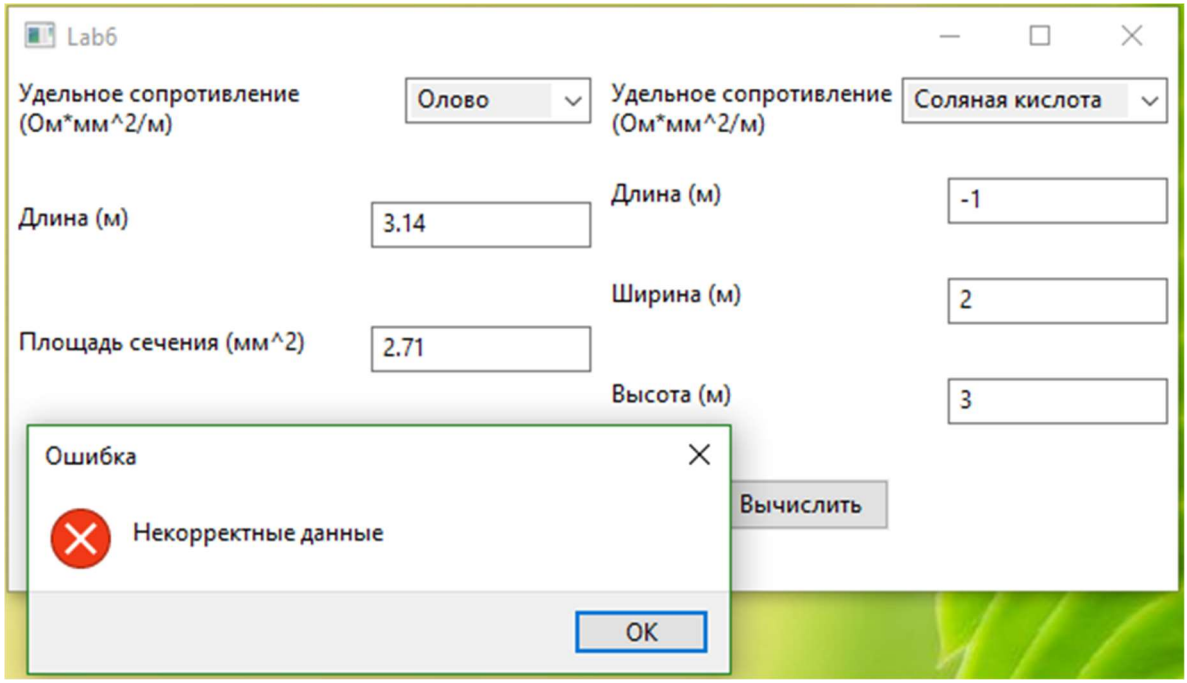
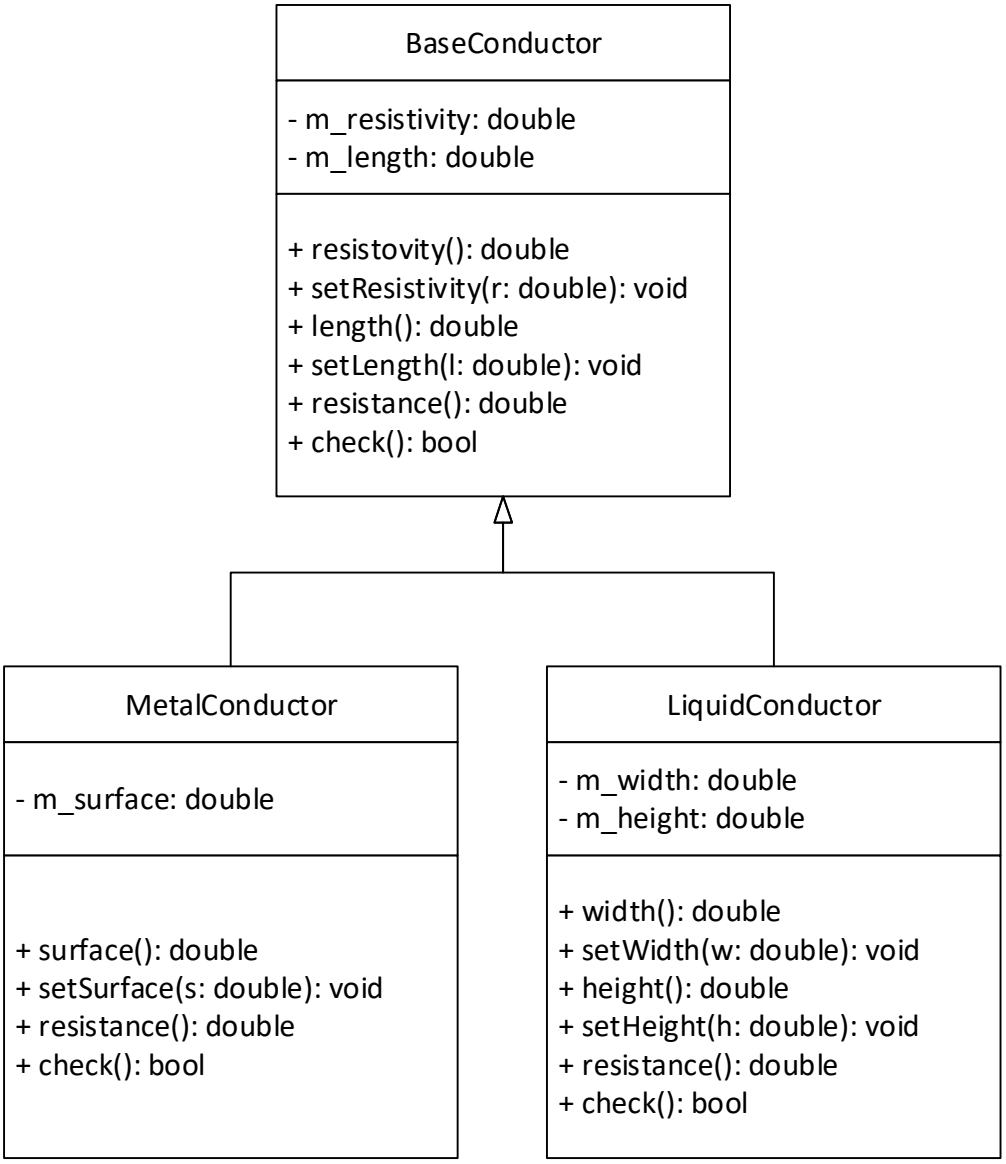


Рисунок 2 – Сообщение о некорректных данных

Приложение Б
(обязательное)
Иерархия классов



Приложение В
(обязательное)
Листинг программы

conductors.h

```
#pragma once

class BaseConductor {
private:
  double m_resistivity = 0;
  double m_length = 0;
public:
  BaseConductor() = default;

  double resistivity() const
  {  return m_resistivity;  }

  void setResistivity(double res)
  {  m_resistivity = res;  }

  double length() const
  { return m_length; }

  void setLength(double length)
  {  m_length = length;  }

  virtual double resistance() const
  {  return 0;  }

  virtual bool check() const
  {  return m_resistivity > 0
        && m_length > 0;  }
};

class   MetalConductor   :   public
BaseConductor {
private:
  double m_surface = 0;
public:
  MetalConductor() = default;

  double surface() const
  {  return m_surface;  }

  void setSurface(double surface)
  {  m_surface = surface;  }

  double resistance() const
  override
  { return resistivity() * length()
        / surface();  }

  bool check() const override
  {  return BaseConductor::check()
        && m_surface > 0;  }
};

class   LiquidConductor   :   public
BaseConductor {
private:
  double m_width = 0;
  double m_heigth = 0;
public:
  LiquidConductor() = default;

  double width() const
  {  return m_width;  }

  void setWidth(double width)
  {  m_width = width;  }

  double height() const
  {  return m_heigth;  }

  void setHeight(double height)
  {  m_heigth = height;  }

  double resistance() const
  override
  {  return resistivity()
        * length()
        / (width() * height()); }

  bool check() const override
  {  return BaseConductor::check()
        && m_width  > 0
        && m_heigth > 0;  }
};
```

main.cpp

```
#include <wx/wx.h>
#include <wx/valnum.h>

#include <map>
#include <memory>

#include "conductors.h"

using fpvalidator = wxFloatingPointValidator<double>;
using stodmap = std::map<wxString, double>;

class MetalSizer : public wxGridSizer {
private:
  BaseConductor* conductor;
  stodmap resistivitymp;
  wxComboBox* resistivityBox;
  wxTextCtrl* lengthEntry;
  wxTextCtrl* surfaceEntry;
  wxButton* compute;

  void OnComputeClick(wxCommandEvent& event) {
    double val;
    bool ok;
    ok = lengthEntry->GetValue().ToDouble(&val);
    conductor->setLength(ok ? val : 0);

    conductor->setResistivity(resistivitymp.at(resistivityBox-
>GetStringSelection()));

    ok = surfaceEntry->GetValue().ToDouble(&val);
    dynamic_cast<MetalConductor*>(conductor)->setSurface(ok ? val : 0);

    if (this->conductor->check()) {
      wxString res = wxString::FromDouble(conductor->resistance());
      wxMessageBox("Сопротивление проводника: " + res + " Ом", "Результат");
    }
    else {
      wxMessageBox("Некорректные данные", "Ошибка", wxICON_ERROR);
    }
  }
public:
  MetalSizer(wxWindow* parent, const stodmap& mr) : wxGridSizer(4, 2, 0, 0),
resistivitymp(mr), conductor(new MetalConductor()) {
    wxSizerFlags textFlags = wxSizerFlags().Left();
    wxSizerFlags boxFlags = wxSizerFlags().Top().Right();

    wxStaticText* resistivity = new wxStaticText(parent, wxID_ANY, "Удельное
сопротивление\n(Ом*мм^2/м)");
    wxStaticText* length = new wxStaticText(parent, wxID_ANY, "Длина (м)");
    wxStaticText* surface = new wxStaticText(parent, wxID_ANY, "Площадь
```

```
сечения (мм^2)");

    resistivity->SetExtraStyle(wxALIGN_LEFT);
    length->SetExtraStyle(wxALIGN_LEFT);
    surface->SetExtraStyle(wxALIGN_LEFT);

    resistivityBox = new wxComboBox(parent, wxID_ANY);
    lengthEntry = new wxTextCtrl(parent, wxID_ANY);
    surfaceEntry = new wxTextCtrl(parent, wxID_ANY);

    lengthEntry->SetValidator(fpvalidator(3, nullptr,
wxNUM_VAL_NO_TRAILING_ZEROES));
    surfaceEntry->SetValidator(fpvalidator(3, nullptr,
wxNUM_VAL_NO_TRAILING_ZEROES));

    resistivityBox->SetEditable(false);
    for (const auto& p : resistivitymp) {
      resistivityBox->AppendString(p.first);
    }
    resistivityBox->SetSelection(0);

    compute = new wxButton(parent, wxID_ANY, "Вычислить");
    compute->Bind(wxEVT_BUTTON, &MetalSizer::OnComputeClick, this);

    this->Add(resistivity, textFlags);
    this->Add(resistivityBox, boxFlags);
    this->Add(length, textFlags);
    this->Add(lengthEntry, boxFlags);
    this->Add(surface, textFlags);
    this->Add(surfaceEntry, boxFlags);
    this->Add(compute, boxFlags);
  }
};

class LiquidSizer : public wxGridSizer {
private:
  BaseConductor* conductor;
  stodmap resistivitymp;
  wxComboBox* resistivityBox;
  wxTextCtrl* lengthEntry;
  wxTextCtrl* widthEntry;
  wxTextCtrl* heightEntry;

  void OnComputeClick(wxCommandEvent& event) {
    double val;
    bool ok;
    ok = lengthEntry->GetValue().ToDouble(&val);
    conductor->setLength(ok ? val : 0);

    ok = widthEntry->GetValue().ToDouble(&val);
```

```
    dynamic_cast<LiquidConductor*>(conductor)->setWidth(ok ? val : 0);

    ok = heightEntry->GetValue().ToDouble(&val);
    dynamic_cast<LiquidConductor*>(conductor)->setHeight(ok ? val : 0);

    conductor->setResistivity(resistivitymp.at(resistivityBox-
>GetStringSelection()));

    if (this->conductor->check()) {
      wxString res = wxString::FromDouble(conductor->resistance());
      wxMessageBox("Сопротивление проводника: " + res + " Ом", "Результат");
    }
    else {
      wxMessageBox("Некорректные данные", "Ошибка", wxICON_ERROR);
    }
  }
public:
  LiquidSizer(wxWindow* parent, const stodmap& lr) : wxGridSizer(5, 2, 0, 0),
resistivitymp(lr), conductor(new LiquidConductor()) {
    wxSizerFlags textFlags = wxSizerFlags().Left();
    wxSizerFlags boxFlags = wxSizerFlags().Top().Right();

    wxStaticText* resistivity = new wxStaticText(parent, wxID_ANY, "Удельное
сопротивление\n(Ом*мм^2/м)");
    wxStaticText* length = new wxStaticText(parent, wxID_ANY, "Длина (м)");
    wxStaticText* width = new wxStaticText(parent, wxID_ANY, "Ширина (м)");
    wxStaticText* height = new wxStaticText(parent, wxID_ANY, "Высота (м)");
    for (wxStaticText* text : { resistivity, length, width, height }) {
      text->SetExtraStyle(wxALIGN_LEFT);
    }

    resistivityBox = new wxComboBox(parent, wxID_ANY);
    lengthEntry = new wxTextCtrl(parent, wxID_ANY);
    widthEntry = new wxTextCtrl(parent, wxID_ANY);
    heightEntry = new wxTextCtrl(parent, wxID_ANY);

    lengthEntry->SetValidator(fpvalidator(3, nullptr,
wxNUM_VAL_NO_TRAILING_ZEROES));
    widthEntry->SetValidator(fpvalidator(3, nullptr,
wxNUM_VAL_NO_TRAILING_ZEROES));
    heightEntry->SetValidator(fpvalidator(3, nullptr,
wxNUM_VAL_NO_TRAILING_ZEROES));

    resistivityBox->SetEditable(false);
    for (const auto& p : resistivitymp) {
      resistivityBox->AppendString(p.first);
    }
    resistivityBox->SetSelection(0);

    wxButton* compute = new wxButton(parent, wxID_ANY, "Вычислить");
```

```
    compute->Bind(wxEVT_BUTTON, &LiquidSizer::OnComputeClick, this);

    this->Add(resistivity, textFlags);
    this->Add(resistivityBox, boxFlags);
    this->Add(length, textFlags);
    this->Add(lengthEntry, boxFlags);
    this->Add(width, textFlags);
    this->Add(widthEntry, boxFlags);
    this->Add(height, textFlags);
    this->Add(heightEntry, boxFlags);
    this->Add(compute, boxFlags);
  }
};

class MainFrame : public wxFrame {
public:
  MainFrame() : wxFrame(NULL, wxID_ANY, "Lab6") {
    stodmap mr = {
      { "Алюминий", 0.028  }, { "Бронза",   0.1    },
      { "Железо",   0.1    }, { "Золото",   0.023  },
      { "Латунь",   0.025  }, { "Медь",     0.0175 },
      { "Олово",    0.12   }, { "Платина",  0.107  },
      { "Свинец",   0.22   }, { "Серебро",  0.015  },
      { "Цинк",     0.054  }, { "Ртуть",    0.94   },
      { "Вольфрам", 0.05   }
    };
    stodmap lr = {
      { "Гидроксид натрия ", 0.032 },  { "Медный купорос",    0.315 },
      { "Серная кислота",    0.025 },  { "Соляная кислота",   0.016 },
      { "Хлорид натрия",     0.083 }
    };
    wxBoxSizer* sizer = new wxBoxSizer(wxHORIZONTAL);
    wxSizerFlags flags = wxSizerFlags().Expand().Border();
    sizer->Add(new MetalSizer(this, mr), flags);
    sizer->Add(new LiquidSizer(this, lr), flags);
    this->SetSizer(sizer);
    this->SetSize(wxSize(600, 300));
    this->SetBackgroundColour(wxColor("White"));
  }
};

class Lab6App : public wxApp {
public:
  virtual bool OnInit() {
    MainFrame *frame = new MainFrame();
    frame->Show(true);
    return true;
  }
};
wxIMPLEMENT_APP(Lab6App);
```