Національний університет «Львівська політехніка»

Кафедра програмного забезпечення

**КУРСОВА РОБОТА**

**з дисципліни «Об’єктно-орієнтоване програмування»**

**На тему:  
*«Розкрій прямокутної коробки»***

Студентки групи  ПЗ-23

спеціальності 6.121

«Інженерія програмного забезпечення»

Кохман О.В.

Керівник: доцент кафедри ПЗ,

к.т.н., доцент Коротєєва Т. О.

Національна шкала \_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів\_\_\_Оцінка ECTS\_\_\_

Члени комісії          \_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Львів 2022

ЗАВДАННЯ

до курсової роботи з дисципліни «Об’єктно-орієнтоване програмування»

студентки групи ПЗ-23 Кохман Олесі

***Тема: «Розкрій прямокутної коробки»***

**Завдання:** (Варіант 11)

Створити таблицю у візуальному середовищі - Матеріал | Ціна за 1м.кв.

1) Розробити розкрій прямокутної коробки, що склеюється. Вивести засобами візуального середовища у вікно малюнок розкрою коробки.

2) За заданими розмірами (ширина, довжина, висота) та матеріалом вирахувати собівартість матеріалів на виготовлення коробки.

3) Вирахувати витрати на комунальні послуги, зарплату, податки за місяць.

4) Визначити собівартість коробки, додати 10% доходу, враховуючи амортизацію обладнання 13,567%.

5) Відсортувати записи таблиці алгоритмом простої вибірки за показником Ціна за 1 м.кв.

6) Визначити сумарний місячний дохід при заданих замовленнях (розмір – матеріал – кількість)

Для [клас](http://vns.lpnu.ua/mod/glossary/showentry.php?eid=54317&displayformat=dictionary)у створити: 1) [Конструктор](http://vns.lpnu.ua/mod/glossary/showentry.php?eid=54330&displayformat=dictionary) за замовчуванням; 2) [Конструктор](http://vns.lpnu.ua/mod/glossary/showentry.php?eid=54330&displayformat=dictionary) з параметрами; 3) [конструктор](http://vns.lpnu.ua/mod/glossary/showentry.php?eid=54330&displayformat=dictionary) копій; 4) перевизначити операції >>, << для зчитування та запису у файл.

**Зміст завдання та календарний план його виконання**

|  |
| --- |
|  |
| **№ з/п** | **Зміст завдання** | **Дата** |
| 1 | Здійснити аналiтичний огляд лiтератури за заданою темою та обгрунтувати вибір інструментальних засобів реалізації. | 12.09.2022 |
| 2 | Побудова UML діаграм | 28.10.2022 |
| 3 | Розробка алгоритмів реалізації | 29.10.2022 |
| 4 | Реалізація завдання (кодування) | 14.11.2022 |
| 5 | Формування інструкції користувача | 20.11.2022 |
| 6 | Оформлення звіту до курсової роботи згідно із вимогами Міжнародних стандартів, дотримуючись такої структури:   * зміст; * алгоритм розв‘язку задачі у покроковому представленні; * діаграми UML класів, прецедентів, послідовності виконання; * код розробленої програми з коментарями; * протокол роботи програми для кожного пункту завдання * інструкція користувача та системні вимоги; * опис виняткових ситуацій; * структура файлу вхідних даних; * висновки; * список використаних джерел. | 22.11.2022 |

Завдання прийнято до виконання: \_\_\_\_\_\_\_\_КОВ\_\_\_\_\_\_\_\_ (Кохман О.В.)

Керівник роботи: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ Коротєєва Т. О./

**Зміст**

1. [Алгоритм розв'язку задачі у покроковому представленні 5](file:///C:\university\git\3-sem\Object-Oriented%20Programming\kursova\Ломницька_ПЗ_23_Курсова_Робота.docx#_Toc117690035)
2. [Діаграми UML класів, прецедентів, послідовності виконання 11](file:///C:\university\git\3-sem\Object-Oriented%20Programming\kursova\Ломницька_ПЗ_23_Курсова_Робота.docx#_Toc117690036)
3. [Код розробленої програми з коментарями 14](file:///C:\university\git\3-sem\Object-Oriented%20Programming\kursova\Ломницька_ПЗ_23_Курсова_Робота.docx#_Toc117690037)
4. [Протокол роботи для кожного пункту завдання 48](file:///C:\university\git\3-sem\Object-Oriented%20Programming\kursova\Ломницька_ПЗ_23_Курсова_Робота.docx#_Toc117690038)
5. [Інструкція користувача та системні вимоги 58](file:///C:\university\git\3-sem\Object-Oriented%20Programming\kursova\Ломницька_ПЗ_23_Курсова_Робота.docx#_Toc117690039)
6. [Опис виняткових ситуацій 68](file:///C:\university\git\3-sem\Object-Oriented%20Programming\kursova\Ломницька_ПЗ_23_Курсова_Робота.docx#_Toc117690040)
7. [Структура файлу вхідних даних 71](file:///C:\university\git\3-sem\Object-Oriented%20Programming\kursova\Ломницька_ПЗ_23_Курсова_Робота.docx#_Toc117690041)

[Висновки 72](file:///C:\university\git\3-sem\Object-Oriented%20Programming\kursova\Ломницька_ПЗ_23_Курсова_Робота.docx#_Toc117690042)

[Список використаних джерел 73](file:///C:\university\git\3-sem\Object-Oriented%20Programming\kursova\Ломницька_ПЗ_23_Курсова_Робота.docx#_Toc117690043)

1. **Алгоритм розв‘язку задачі у покроковому представленні**

Алгоритм розв’язку задачі загалом:

**Алгоритм General:**

**G1:** зчитати дані з файлу або зз клавіатури.

**G2:** виконати необхідні дії.

**G3:** зберегти дані у файл.

Алгоритм додавання даних з клавіатури:

**Алгоритм Add:**

Задано ім’я матеріалу та ціну, створений лінійний список типу Material\*, створена таблиця для запису даних.

**A1:** спробувати виконати кроки A2,A3,A4,A5, в іншому випадку перейти до A7.

**A2:** динамічно створити новий об’єкт типу Material, де помістити ім’я матеріалу та ціну.

**A3:** створити новий рядок у таблиці.

**A4**: додати об’єкт у таблицю.

**A5:** додати об’єкт у лінійний список.

**A6:** вихід.

**A7:** виведення повідомлення про помилку, вихід.

Алгоритм додавання даних з файлу:

**Алгоритм AddFromFile:**

Задано ім’я матеріалу та ціну, створений лінійний список типу Material\*, створена таблиця для запису даних.

**AFF1:** перевірити умову чи файл є відкритий, якщо так , то виконати кроки

AFF2, AFF3, AFF4, AFF5, AFF6, AFF7, якщо ні, то крок AFF8.

**AFF2:** створити новий об’єкт типу Material

**AFF3:** вхід у цикл, який виконується поки не закінчиться файл. Виконати кроки AFF4, AFF5, AFF6, AFF7.

**AFF4:** створити новий рядок у таблиці

**AFF5:** додати дані у таблицю.

**AFF6:** додати дані у список, створити новий об’єкт типу Material.

**AFF7:** закрити файл, вихід.

**AFF8:** вивести повідомлення про помилку, вихід

Алгоритм зберігання у файл:

**Алгоритм Save:**

Задано файл, у який буде записуватись інформація, лінійних список типу Material\*.

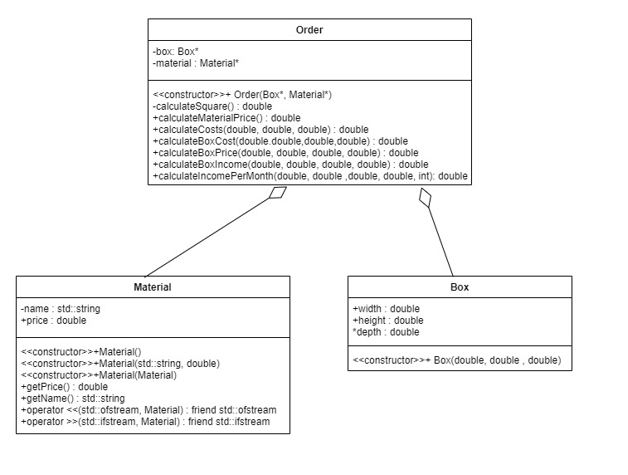
S1: відкрити файл з параметром trunc.

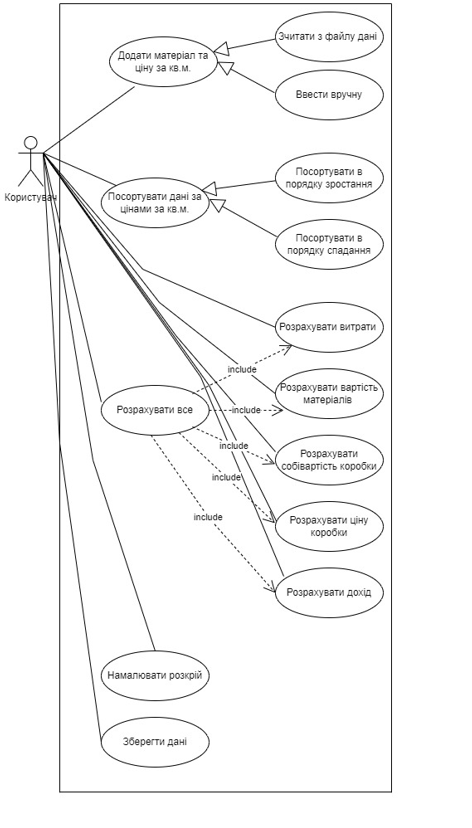
S2: цикл, який присвоює ітератору вказівник на початок списку, триває поки ітератор не дорівнюватиме кінцю списку і виконуватиме крок S3.

S3: запис даних у файл.

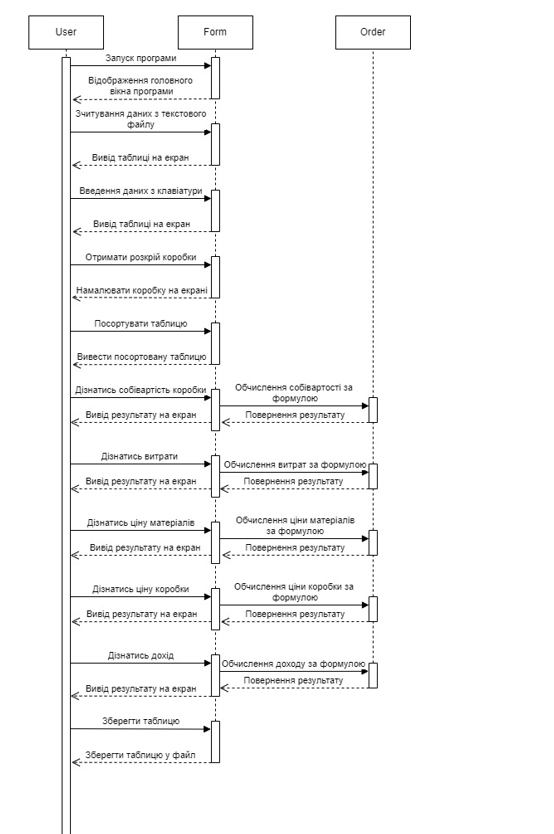
S4: закриття файлу, вихід.

1. **Діаграми UML класів, прецедентів, послідовності виконання**

*****Рис. 2.1. Діаграма класів рівня реалізації.*

******

*Рис. 2.2. Діаграма прецедентів.*

**

*Рис. 2.3. Діаграма послідовності.*

1. **Код розробленої програми з коментарями**

Назва файлу: Box.h

#ifndef BOX\_H

#define BOX\_H

#pragma once

class Box { // клас для боксу

public:

double width;

double height;

double depth;

Box(double width, double height, double depth);

};

#endif

Назва файлу: Box.cpp

#include "Box.h"

Box::Box(double width, double height, double depth) {

this->width = width;

this->height = height;

this->depth = depth;

}

Назва файлу: Material.h

#ifndef MATERIAL\_H

#define MATERIAL\_H

#pragma once

#include <string>

#include <fstream>

class RowIsNotSelectedException : public std::exception { // виняток,, якщо рядок не вибраний

};

class MaterialsAreEmptyException : public std::exception { //виняток, якщо ліст матеріалів пустий

};

class Material { // матеріал боксу

private:

std::string name;

public:

std::string getName();

double getPrice();

double price;

Material(); // конструктор за замовчуванням

Material(std::string name, double price); // конструктор з параметрами

Material(Material& material); // конструктор копіювання

friend std::ofstream& operator << (std::ofstream& file, Material\* material); // перевантажена операція запису в файл

friend std::ifstream& operator >> (std::ifstream& file, Material\* material); // перевантажена операція зчитування з файлу

bool operator <(const Material& material) const; // перевантажений оператор порівняння

};

#endif

Назва файлу: Material.cpp

#include "Material.h"

Material::Material() {

this->name = "Wood";

this->price = 10;

}

Material::Material(std::string name, double price) {

this->name = name;

this->price = price;

}

bool Material::operator < (const Material& material) const

{

return this->price < material.price;

}

std::string Material::getName() {

return this->name;

}

double Material::getPrice() {

return this->price;

}

std::ofstream& operator << (std::ofstream& file, Material\* material) {

file << (material->getName() + " " + std::to\_string(material->getPrice()) + '\n');

return file;

}

std::ifstream& operator >> (std::ifstream& file, Material\* material) {

file >> material->name;

file >> material->price;

return file;

}

Material::Material(Material& material) {

name = material.name;

price = material.price;

}

Назва файлу: Order.h

#ifndef ORDER\_H

#define ORDER\_H

#pragma once

#include "Box.h"

#include "Material.h"

class Order { // клас-замовлення

private:

Box\* box;

Material\* material;

double calculateSquare();

public:

Order(Box\* box, Material\* material);

double calculateMaterialPrice();

double calculateCosts(double utilityRate, double salaryRate, double taxRate);

double calculateBoxCost(double utilityRate, double salaryRate, double taxRate, double equipmentDeprecationPrice);

double calculateBoxPrice(double utilityRate, double salaryRate, double taxRate, double equipmentDeprecationPrice);

double calculateBoxIncome(double utilityRate, double salaryRate, double taxRate, double equipmentDeprecationPrice);

double calculateIncomePerMonth(double utilityRate, double salaryRate, double taxRate, double equipmentDeprecationPrice, int countOfBoxes);

};

#endif

Назва файлу:Order.cpp

#include "Order.h"

Order::Order(Box\* box, Material\* material) {

this->box = box;

this->material = material;

}

double Order::calculateSquare() {

double topAndBottom = 2 \* (box->width \* box->depth);

double walls = 4 \* box->depth \* box->height;

double wallsFastening = box->height \* (double)box->height / 10.0;

double leftSmallSide = box->depth \* (double)box->width / 5.2;

return (topAndBottom + walls + wallsFastening + leftSmallSide) / 100.0;

} // вираховує площу

double Order::calculateMaterialPrice() { // підраховує ціну матеріалу

return calculateSquare() \* material->getPrice();

}

double Order::calculateCosts(double utilityRate, double salaryRate, double taxRate) { // підраховує витрати на комунальні послуги, податки та зарплату

return calculateSquare() \* (utilityRate + salaryRate + taxRate);

}

double Order::calculateBoxCost(double utilityRate, double salaryRate, double taxRate, double equipmentDeprecationPrice) {

return calculateCosts(utilityRate, salaryRate, taxRate) + calculateMaterialPrice() + equipmentDeprecationPrice \* 0.13567;

} // підраховує ціну коробки в залежності від комунальних послуг, податків, зарплати та амортизації обладнання

double Order::calculateBoxPrice(double utilityRate, double salaryRate, double taxRate, double equipmentDeprecationPrice) {

return calculateBoxCost(utilityRate, salaryRate, taxRate, equipmentDeprecationPrice) \* 1.1;

}

// підраховує ціну коробки для клієнта

double Order::calculateBoxIncome(double utilityRate, double salaryRate, double taxRate, double equipmentDeprecationPrice) {

return calculateBoxPrice(utilityRate, salaryRate, taxRate, equipmentDeprecationPrice) \* 0.1;

} // підраховує дохід

double Order::calculateIncomePerMonth(double utilityRate, double salaryRate, double taxRate, double equipmentDeprecationPrice, int countOfBoxes) {

return calculateBoxIncome(utilityRate, salaryRate, taxRate, equipmentDeprecationPrice) \* countOfBoxes;

} // підраховує дохід за місяць

Назва файлу: MyForm.h

#pragma once

#include "Material.h"

#include <list>

#include "Box.h"

#include "Order.h"

#include <iostream>

namespace kursova {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

/// <summary>

/// Summary for MyForm

/// </summary>

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

{

public:

MyForm(void)

{

InitializeComponent();

//

//TODO: Add the constructor code here

//

}

protected:

/// <summary>

/// Clean up any resources being used.

/// </summary>

~MyForm()

{

if (components)

{

delete components;

}

}

private:

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

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

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

private: System::Windows::Forms::Button^ drawButton;

protected:

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

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

private: System::Windows::Forms::Button^ addButton;

private: System::Windows::Forms::Button^ importMaterialsButton;

private: System::Windows::Forms::DataGridView^ materialsDataGridView;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ materialName;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ price;

private: System::Windows::Forms::Button^ saveButton;

private: System::Windows::Forms::Button^ sortAscbutton;

private: System::Windows::Forms::Button^ sortDescbutton;

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

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

private: System::Windows::Forms::PictureBox^ drawBox;

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

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

private: System::Windows::Forms::Label^ label5;

private: System::Windows::Forms::Button^ calculateMaterialPriceButton;

private: System::Windows::Forms::Label^ label6;

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

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

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

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

private: System::Windows::Forms::Label^ label7;

private: System::Windows::Forms::Label^ label8;

private: System::Windows::Forms::Label^ label9;

private: System::Windows::Forms::Button^ calculateCostsButton;

private: System::Windows::Forms::Label^ label10;

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

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

private: System::Windows::Forms::Label^ label11;

private: System::Windows::Forms::Button^ calculateBoxCostButton;

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

private: System::Windows::Forms::Label^ label13;

private: System::Windows::Forms::Button^ calculateBoxPriceButton;

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

private: System::Windows::Forms::Label^ label14;

private: System::Windows::Forms::Label^ label12;

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

private: System::Windows::Forms::Button^ calculateIncomeButton;

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

private: System::Windows::Forms::Label^ label16;

private: System::Windows::Forms::Button^ calculateAllbutton;

private:

/// <summary>

/// Required designer variable.

/// </summary>

System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code

/// <summary>

/// Required method for Designer support - do not modify

/// the contents of this method with the code editor.

/// </summary>

void InitializeComponent(void)

{

this->boxWidthTextBox = (gcnew System::Windows::Forms::TextBox());

this->boxHeightTextBox = (gcnew System::Windows::Forms::TextBox());

this->boxDepthTextBox = (gcnew System::Windows::Forms::TextBox());

this->drawButton = (gcnew System::Windows::Forms::Button());

this->nameTextBox = (gcnew System::Windows::Forms::TextBox());

this->priceTextBox = (gcnew System::Windows::Forms::TextBox());

this->addButton = (gcnew System::Windows::Forms::Button());

this->importMaterialsButton = (gcnew System::Windows::Forms::Button());

this->materialsDataGridView = (gcnew System::Windows::Forms::DataGridView());

this->materialName = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->price = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->saveButton = (gcnew System::Windows::Forms::Button());

this->sortAscbutton = (gcnew System::Windows::Forms::Button());

this->sortDescbutton = (gcnew System::Windows::Forms::Button());

this->label1 = (gcnew System::Windows::Forms::Label());

this->label2 = (gcnew System::Windows::Forms::Label());

this->drawBox = (gcnew System::Windows::Forms::PictureBox());

this->label3 = (gcnew System::Windows::Forms::Label());

this->label4 = (gcnew System::Windows::Forms::Label());

this->label5 = (gcnew System::Windows::Forms::Label());

this->calculateMaterialPriceButton = (gcnew System::Windows::Forms::Button());

this->label6 = (gcnew System::Windows::Forms::Label());

this->materialPriceLabel = (gcnew System::Windows::Forms::Label());

this->utilityRateTextBox = (gcnew System::Windows::Forms::TextBox());

this->salaryRateTextBox = (gcnew System::Windows::Forms::TextBox());

this->taxRateTextBox = (gcnew System::Windows::Forms::TextBox());

this->label7 = (gcnew System::Windows::Forms::Label());

this->label8 = (gcnew System::Windows::Forms::Label());

this->label9 = (gcnew System::Windows::Forms::Label());

this->calculateCostsButton = (gcnew System::Windows::Forms::Button());

this->label10 = (gcnew System::Windows::Forms::Label());

this->costsLabel = (gcnew System::Windows::Forms::Label());

this->equipmentDeprecationPriceTextBox = (gcnew System::Windows::Forms::TextBox());

this->label11 = (gcnew System::Windows::Forms::Label());

this->calculateBoxCostButton = (gcnew System::Windows::Forms::Button());

this->boxCostLabel = (gcnew System::Windows::Forms::Label());

this->label13 = (gcnew System::Windows::Forms::Label());

this->calculateBoxPriceButton = (gcnew System::Windows::Forms::Button());

this->boxPriceLabel = (gcnew System::Windows::Forms::Label());

this->label14 = (gcnew System::Windows::Forms::Label());

this->label12 = (gcnew System::Windows::Forms::Label());

this->countOfBoxesPerMonthTextBox = (gcnew System::Windows::Forms::TextBox());

this->calculateIncomeButton = (gcnew System::Windows::Forms::Button());

this->incomePerMonthLabel = (gcnew System::Windows::Forms::Label());

this->label16 = (gcnew System::Windows::Forms::Label());

this->calculateAllbutton = (gcnew System::Windows::Forms::Button());

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->materialsDataGridView))->BeginInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->drawBox))->BeginInit();

this->SuspendLayout();

//

// boxWidthTextBox

//

this->boxWidthTextBox->AccessibleDescription = L"";

this->boxWidthTextBox->AccessibleName = L"";

this->boxWidthTextBox->BackColor = System::Drawing::SystemColors::Window;

this->boxWidthTextBox->BorderStyle = System::Windows::Forms::BorderStyle::FixedSingle;

this->boxWidthTextBox->ForeColor = System::Drawing::SystemColors::WindowText;

this->boxWidthTextBox->Location = System::Drawing::Point(16, 508);

this->boxWidthTextBox->Margin = System::Windows::Forms::Padding(4);

this->boxWidthTextBox->Name = L"boxWidthTextBox";

this->boxWidthTextBox->Size = System::Drawing::Size(132, 22);

this->boxWidthTextBox->TabIndex = 0;

this->boxWidthTextBox->Tag = L"";

//

// boxHeightTextBox

//

this->boxHeightTextBox->BackColor = System::Drawing::SystemColors::Window;

this->boxHeightTextBox->ForeColor = System::Drawing::SystemColors::WindowText;

this->boxHeightTextBox->Location = System::Drawing::Point(16, 555);

this->boxHeightTextBox->Margin = System::Windows::Forms::Padding(4);

this->boxHeightTextBox->Name = L"boxHeightTextBox";

this->boxHeightTextBox->Size = System::Drawing::Size(132, 22);

this->boxHeightTextBox->TabIndex = 1;

//

// boxDepthTextBox

//

this->boxDepthTextBox->BackColor = System::Drawing::SystemColors::Window;

this->boxDepthTextBox->Location = System::Drawing::Point(16, 603);

this->boxDepthTextBox->Margin = System::Windows::Forms::Padding(4);

this->boxDepthTextBox->Name = L"boxDepthTextBox";

this->boxDepthTextBox->Size = System::Drawing::Size(132, 22);

this->boxDepthTextBox->TabIndex = 2;

//

// drawButton

//

this->drawButton->BackColor = System::Drawing::SystemColors::ActiveCaption;

this->drawButton->FlatStyle = System::Windows::Forms::FlatStyle::Popup;

this->drawButton->Location = System::Drawing::Point(16, 635);

this->drawButton->Margin = System::Windows::Forms::Padding(4);

this->drawButton->Name = L"drawButton";

this->drawButton->Size = System::Drawing::Size(133, 28);

this->drawButton->TabIndex = 3;

this->drawButton->Text = L"Draw";

this->drawButton->UseVisualStyleBackColor = false;

this->drawButton->Click += gcnew System::EventHandler(this, &MyForm::button1\_Click);

//

// nameTextBox

//

this->nameTextBox->Location = System::Drawing::Point(1095, 507);

this->nameTextBox->Margin = System::Windows::Forms::Padding(4);

this->nameTextBox->Name = L"nameTextBox";

this->nameTextBox->Size = System::Drawing::Size(191, 22);

this->nameTextBox->TabIndex = 4;

//

// priceTextBox

//

this->priceTextBox->Location = System::Drawing::Point(1310, 507);

this->priceTextBox->Margin = System::Windows::Forms::Padding(4);

this->priceTextBox->Name = L"priceTextBox";

this->priceTextBox->Size = System::Drawing::Size(191, 22);

this->priceTextBox->TabIndex = 5;

//

// addButton

//

this->addButton->FlatStyle = System::Windows::Forms::FlatStyle::Popup;

this->addButton->Location = System::Drawing::Point(1095, 549);

this->addButton->Margin = System::Windows::Forms::Padding(4);

this->addButton->Name = L"addButton";

this->addButton->Size = System::Drawing::Size(191, 28);

this->addButton->TabIndex = 6;

this->addButton->Text = L"Add";

this->addButton->UseVisualStyleBackColor = true;

this->addButton->Click += gcnew System::EventHandler(this, &MyForm::addButton\_Click);

//

// importMaterialsButton

//

this->importMaterialsButton->FlatStyle = System::Windows::Forms::FlatStyle::Popup;

this->importMaterialsButton->Location = System::Drawing::Point(1310, 549);

this->importMaterialsButton->Margin = System::Windows::Forms::Padding(4);

this->importMaterialsButton->Name = L"importMaterialsButton";

this->importMaterialsButton->Size = System::Drawing::Size(191, 28);

this->importMaterialsButton->TabIndex = 7;

this->importMaterialsButton->Text = L"Import materials";

this->importMaterialsButton->UseVisualStyleBackColor = true;

this->importMaterialsButton->Click += gcnew System::EventHandler(this, &MyForm::importMaterialsButton\_Click);

//

// materialsDataGridView

//

this->materialsDataGridView->AllowUserToAddRows = false;

this->materialsDataGridView->AllowUserToDeleteRows = false;

this->materialsDataGridView->AllowUserToResizeColumns = false;

this->materialsDataGridView->AllowUserToResizeRows = false;

this->materialsDataGridView->BackgroundColor = System::Drawing::SystemColors::ActiveCaption;

this->materialsDataGridView->ColumnHeadersHeightSizeMode = System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;

this->materialsDataGridView->Columns->AddRange(gcnew cli::array< System::Windows::Forms::DataGridViewColumn^ >(2) {

this->materialName,

this->price

});

this->materialsDataGridView->GridColor = System::Drawing::SystemColors::ActiveCaption;

this->materialsDataGridView->Location = System::Drawing::Point(1095, 15);

this->materialsDataGridView->Margin = System::Windows::Forms::Padding(4);

this->materialsDataGridView->MultiSelect = false;

this->materialsDataGridView->Name = L"materialsDataGridView";

this->materialsDataGridView->ReadOnly = true;

this->materialsDataGridView->RowHeadersWidth = 51;

this->materialsDataGridView->Size = System::Drawing::Size(406, 470);

this->materialsDataGridView->TabIndex = 9;

//

// materialName

//

this->materialName->HeaderText = L"Material Name";

this->materialName->MinimumWidth = 6;

this->materialName->Name = L"materialName";

this->materialName->ReadOnly = true;

this->materialName->Width = 125;

//

// price

//

this->price->HeaderText = L"Price/m^2";

this->price->MinimumWidth = 6;

this->price->Name = L"price";

this->price->ReadOnly = true;

this->price->Width = 125;

//

// saveButton

//

this->saveButton->FlatStyle = System::Windows::Forms::FlatStyle::Popup;

this->saveButton->Location = System::Drawing::Point(1095, 619);

this->saveButton->Margin = System::Windows::Forms::Padding(4);

this->saveButton->Name = L"saveButton";

this->saveButton->Size = System::Drawing::Size(406, 28);

this->saveButton->TabIndex = 11;

this->saveButton->Text = L"Save";

this->saveButton->UseVisualStyleBackColor = true;

this->saveButton->Click += gcnew System::EventHandler(this, &MyForm::saveButton\_Click);

//

// sortAscbutton

//

this->sortAscbutton->FlatStyle = System::Windows::Forms::FlatStyle::Popup;

this->sortAscbutton->Location = System::Drawing::Point(1095, 583);

this->sortAscbutton->Margin = System::Windows::Forms::Padding(4);

this->sortAscbutton->Name = L"sortAscbutton";

this->sortAscbutton->Size = System::Drawing::Size(191, 28);

this->sortAscbutton->TabIndex = 12;

this->sortAscbutton->Text = L"Sort asc";

this->sortAscbutton->UseVisualStyleBackColor = true;

this->sortAscbutton->Click += gcnew System::EventHandler(this, &MyForm::sortAscbutton\_Click);

//

// sortDescbutton

//

this->sortDescbutton->FlatStyle = System::Windows::Forms::FlatStyle::Popup;

this->sortDescbutton->Location = System::Drawing::Point(1310, 585);

this->sortDescbutton->Margin = System::Windows::Forms::Padding(4);

this->sortDescbutton->Name = L"sortDescbutton";

this->sortDescbutton->Size = System::Drawing::Size(191, 28);

this->sortDescbutton->TabIndex = 13;

this->sortDescbutton->Text = L"Sort desc";

this->sortDescbutton->UseVisualStyleBackColor = true;

this->sortDescbutton->Click += gcnew System::EventHandler(this, &MyForm::sortDescbutton\_Click);

//

// label1

//

this->label1->AutoSize = true;

this->label1->Location = System::Drawing::Point(1169, 490);

this->label1->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->label1->Name = L"label1";

this->label1->Size = System::Drawing::Size(44, 16);

this->label1->TabIndex = 14;

this->label1->Text = L"Name";

//

// label2

//

this->label2->AutoSize = true;

this->label2->Location = System::Drawing::Point(1382, 490);

this->label2->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->label2->Name = L"label2";

this->label2->Size = System::Drawing::Size(38, 16);

this->label2->TabIndex = 15;

this->label2->Text = L"Price";

//

// drawBox

//

this->drawBox->BackColor = System::Drawing::SystemColors::ActiveCaption;

this->drawBox->Location = System::Drawing::Point(16, 15);

this->drawBox->Margin = System::Windows::Forms::Padding(4);

this->drawBox->Name = L"drawBox";

this->drawBox->Size = System::Drawing::Size(1071, 470);

this->drawBox->TabIndex = 16;

this->drawBox->TabStop = false;

this->drawBox->Paint += gcnew System::Windows::Forms::PaintEventHandler(this, &MyForm::drawBox\_Paint);

//

// label3

//

this->label3->AutoSize = true;

this->label3->Location = System::Drawing::Point(16, 489);

this->label3->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->label3->Name = L"label3";

this->label3->Size = System::Drawing::Size(41, 16);

this->label3->TabIndex = 17;

this->label3->Text = L"Width";

//

// label4

//

this->label4->AutoSize = true;

this->label4->Location = System::Drawing::Point(16, 535);

this->label4->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->label4->Name = L"label4";

this->label4->Size = System::Drawing::Size(46, 16);

this->label4->TabIndex = 18;

this->label4->Text = L"Height";

//

// label5

//

this->label5->AutoSize = true;

this->label5->Location = System::Drawing::Point(16, 583);

this->label5->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->label5->Name = L"label5";

this->label5->Size = System::Drawing::Size(43, 16);

this->label5->TabIndex = 19;

this->label5->Text = L"Depth";

//

// calculateMaterialPriceButton

//

this->calculateMaterialPriceButton->FlatStyle = System::Windows::Forms::FlatStyle::Popup;

this->calculateMaterialPriceButton->Location = System::Drawing::Point(367, 549);

this->calculateMaterialPriceButton->Margin = System::Windows::Forms::Padding(4);

this->calculateMaterialPriceButton->Name = L"calculateMaterialPriceButton";

this->calculateMaterialPriceButton->Size = System::Drawing::Size(167, 28);

this->calculateMaterialPriceButton->TabIndex = 20;

this->calculateMaterialPriceButton->Text = L"Calculate Material Price";

this->calculateMaterialPriceButton->UseVisualStyleBackColor = true;

this->calculateMaterialPriceButton->Click += gcnew System::EventHandler(this, &MyForm::calculateMaterialPriceButton\_Click);

//

// label6

//

this->label6->AutoSize = true;

this->label6->Location = System::Drawing::Point(364, 597);

this->label6->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->label6->Name = L"label6";

this->label6->Size = System::Drawing::Size(89, 16);

this->label6->TabIndex = 21;

this->label6->Text = L"Material Price";

//

// materialPriceLabel

//

this->materialPriceLabel->AutoSize = true;

this->materialPriceLabel->Location = System::Drawing::Point(432, 606);

this->materialPriceLabel->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->materialPriceLabel->Name = L"materialPriceLabel";

this->materialPriceLabel->Size = System::Drawing::Size(0, 16);

this->materialPriceLabel->TabIndex = 22;

//

// utilityRateTextBox

//

this->utilityRateTextBox->Location = System::Drawing::Point(227, 510);

this->utilityRateTextBox->Margin = System::Windows::Forms::Padding(4);

this->utilityRateTextBox->Name = L"utilityRateTextBox";

this->utilityRateTextBox->Size = System::Drawing::Size(132, 22);

this->utilityRateTextBox->TabIndex = 23;

//

// salaryRateTextBox

//

this->salaryRateTextBox->Location = System::Drawing::Point(367, 510);

this->salaryRateTextBox->Margin = System::Windows::Forms::Padding(4);

this->salaryRateTextBox->Name = L"salaryRateTextBox";

this->salaryRateTextBox->Size = System::Drawing::Size(167, 22);

this->salaryRateTextBox->TabIndex = 24;

//

// taxRateTextBox

//

this->taxRateTextBox->Location = System::Drawing::Point(542, 510);

this->taxRateTextBox->Margin = System::Windows::Forms::Padding(4);

this->taxRateTextBox->Name = L"taxRateTextBox";

this->taxRateTextBox->Size = System::Drawing::Size(167, 22);

this->taxRateTextBox->TabIndex = 25;

//

// label7

//

this->label7->AutoSize = true;

this->label7->Location = System::Drawing::Point(234, 490);

this->label7->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->label7->Name = L"label7";

this->label7->Size = System::Drawing::Size(106, 16);

this->label7->TabIndex = 26;

this->label7->Text = L"Utility Rate / m^2";

//

// label8

//

this->label8->AutoSize = true;

this->label8->Location = System::Drawing::Point(388, 490);

this->label8->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->label8->Name = L"label8";

this->label8->Size = System::Drawing::Size(113, 16);

this->label8->TabIndex = 27;

this->label8->Text = L"Salary Rate / m^2";

//

// label9

//

this->label9->AutoSize = true;

this->label9->Location = System::Drawing::Point(562, 490);

this->label9->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->label9->Name = L"label9";

this->label9->Size = System::Drawing::Size(97, 16);

this->label9->TabIndex = 28;

this->label9->Text = L"Tax Rate / m^2";

//

// calculateCostsButton

//

this->calculateCostsButton->BackColor = System::Drawing::SystemColors::ActiveCaption;

this->calculateCostsButton->FlatStyle = System::Windows::Forms::FlatStyle::Popup;

this->calculateCostsButton->Location = System::Drawing::Point(227, 549);

this->calculateCostsButton->Margin = System::Windows::Forms::Padding(4);

this->calculateCostsButton->Name = L"calculateCostsButton";

this->calculateCostsButton->Size = System::Drawing::Size(132, 28);

this->calculateCostsButton->TabIndex = 29;

this->calculateCostsButton->Text = L"Calculate Costs";

this->calculateCostsButton->UseVisualStyleBackColor = false;

this->calculateCostsButton->Click += gcnew System::EventHandler(this, &MyForm::calculateCostsButton\_Click);

//

// label10

//

this->label10->AutoSize = true;

this->label10->Location = System::Drawing::Point(224, 595);

this->label10->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->label10->Name = L"label10";

this->label10->Size = System::Drawing::Size(41, 16);

this->label10->TabIndex = 30;

this->label10->Text = L"Costs";

//

// costsLabel

//

this->costsLabel->AutoSize = true;

this->costsLabel->Location = System::Drawing::Point(224, 606);

this->costsLabel->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->costsLabel->Name = L"costsLabel";

this->costsLabel->Size = System::Drawing::Size(0, 16);

this->costsLabel->TabIndex = 31;

//

// equipmentDeprecationPriceTextBox

//

this->equipmentDeprecationPriceTextBox->Location = System::Drawing::Point(717, 510);

this->equipmentDeprecationPriceTextBox->Margin = System::Windows::Forms::Padding(4);

this->equipmentDeprecationPriceTextBox->Name = L"equipmentDeprecationPriceTextBox";

this->equipmentDeprecationPriceTextBox->Size = System::Drawing::Size(191, 22);

this->equipmentDeprecationPriceTextBox->TabIndex = 32;

//

// label11

//

this->label11->AutoSize = true;

this->label11->Location = System::Drawing::Point(714, 490);

this->label11->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->label11->Name = L"label11";

this->label11->Size = System::Drawing::Size(182, 16);

this->label11->TabIndex = 33;

this->label11->Text = L"Equipment depreciation price";

//

// calculateBoxCostButton

//

this->calculateBoxCostButton->FlatStyle = System::Windows::Forms::FlatStyle::Popup;

this->calculateBoxCostButton->Location = System::Drawing::Point(542, 549);

this->calculateBoxCostButton->Margin = System::Windows::Forms::Padding(4);

this->calculateBoxCostButton->Name = L"calculateBoxCostButton";

this->calculateBoxCostButton->Size = System::Drawing::Size(167, 28);

this->calculateBoxCostButton->TabIndex = 34;

this->calculateBoxCostButton->Text = L"Calculate Box Cost";

this->calculateBoxCostButton->UseVisualStyleBackColor = true;

this->calculateBoxCostButton->Click += gcnew System::EventHandler(this, &MyForm::calculateBoxCostButton\_Click);

//

// boxCostLabel

//

this->boxCostLabel->AutoSize = true;

this->boxCostLabel->Location = System::Drawing::Point(591, 606);

this->boxCostLabel->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->boxCostLabel->Name = L"boxCostLabel";

this->boxCostLabel->Size = System::Drawing::Size(0, 16);

this->boxCostLabel->TabIndex = 36;

//

// label13

//

this->label13->AutoSize = true;

this->label13->Location = System::Drawing::Point(539, 597);

this->label13->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->label13->Name = L"label13";

this->label13->Size = System::Drawing::Size(60, 16);

this->label13->TabIndex = 35;

this->label13->Text = L"Box Cost";

//

// calculateBoxPriceButton

//

this->calculateBoxPriceButton->FlatStyle = System::Windows::Forms::FlatStyle::Popup;

this->calculateBoxPriceButton->Location = System::Drawing::Point(717, 549);

this->calculateBoxPriceButton->Margin = System::Windows::Forms::Padding(4);

this->calculateBoxPriceButton->Name = L"calculateBoxPriceButton";

this->calculateBoxPriceButton->Size = System::Drawing::Size(191, 28);

this->calculateBoxPriceButton->TabIndex = 37;

this->calculateBoxPriceButton->Text = L"Calculate Box Price";

this->calculateBoxPriceButton->UseVisualStyleBackColor = true;

this->calculateBoxPriceButton->Click += gcnew System::EventHandler(this, &MyForm::calculateBoxPriceButton\_Click);

//

// boxPriceLabel

//

this->boxPriceLabel->AutoSize = true;

this->boxPriceLabel->Location = System::Drawing::Point(789, 607);

this->boxPriceLabel->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->boxPriceLabel->Name = L"boxPriceLabel";

this->boxPriceLabel->Size = System::Drawing::Size(0, 16);

this->boxPriceLabel->TabIndex = 39;

//

// label14

//

this->label14->AutoSize = true;

this->label14->Location = System::Drawing::Point(725, 597);

this->label14->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->label14->Name = L"label14";

this->label14->Size = System::Drawing::Size(64, 16);

this->label14->TabIndex = 38;

this->label14->Text = L"Box Price";

//

// label12

//

this->label12->AutoSize = true;

this->label12->Location = System::Drawing::Point(913, 490);

this->label12->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->label12->Name = L"label12";

this->label12->Size = System::Drawing::Size(157, 16);

this->label12->TabIndex = 40;

this->label12->Text = L"Count of boxes per month";

//

// countOfBoxesPerMonthTextBox

//

this->countOfBoxesPerMonthTextBox->Location = System::Drawing::Point(916, 510);

this->countOfBoxesPerMonthTextBox->Margin = System::Windows::Forms::Padding(4);

this->countOfBoxesPerMonthTextBox->Name = L"countOfBoxesPerMonthTextBox";

this->countOfBoxesPerMonthTextBox->Size = System::Drawing::Size(163, 22);

this->countOfBoxesPerMonthTextBox->TabIndex = 41;

//

// calculateIncomeButton

//

this->calculateIncomeButton->FlatStyle = System::Windows::Forms::FlatStyle::Popup;

this->calculateIncomeButton->Location = System::Drawing::Point(916, 549);

this->calculateIncomeButton->Margin = System::Windows::Forms::Padding(4);

this->calculateIncomeButton->Name = L"calculateIncomeButton";

this->calculateIncomeButton->Size = System::Drawing::Size(163, 28);

this->calculateIncomeButton->TabIndex = 42;

this->calculateIncomeButton->Text = L"Calculate Income";

this->calculateIncomeButton->UseVisualStyleBackColor = true;

this->calculateIncomeButton->Click += gcnew System::EventHandler(this, &MyForm::calculateIncomeButton\_Click);

//

// incomePerMonthLabel

//

this->incomePerMonthLabel->AutoSize = true;

this->incomePerMonthLabel->Location = System::Drawing::Point(1087, 606);

this->incomePerMonthLabel->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->incomePerMonthLabel->Name = L"incomePerMonthLabel";

this->incomePerMonthLabel->Size = System::Drawing::Size(0, 16);

this->incomePerMonthLabel->TabIndex = 44;

//

// label16

//

this->label16->AutoSize = true;

this->label16->Location = System::Drawing::Point(913, 597);

this->label16->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->label16->Name = L"label16";

this->label16->Size = System::Drawing::Size(113, 16);

this->label16->TabIndex = 43;

this->label16->Text = L"Income per month";

//

// calculateAllbutton

//

this->calculateAllbutton->FlatStyle = System::Windows::Forms::FlatStyle::Flat;

this->calculateAllbutton->Location = System::Drawing::Point(542, 619);

this->calculateAllbutton->Margin = System::Windows::Forms::Padding(4);

this->calculateAllbutton->Name = L"calculateAllbutton";

this->calculateAllbutton->Size = System::Drawing::Size(167, 28);

this->calculateAllbutton->TabIndex = 45;

this->calculateAllbutton->Text = L"Calculate All";

this->calculateAllbutton->UseVisualStyleBackColor = true;

this->calculateAllbutton->Click += gcnew System::EventHandler(this, &MyForm::calculateAllbutton\_Click);

//

// MyForm

//

this->AutoScaleDimensions = System::Drawing::SizeF(8, 16);

this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;

this->BackColor = System::Drawing::SystemColors::ActiveCaption;

this->ClientSize = System::Drawing::Size(1565, 673);

this->Controls->Add(this->calculateAllbutton);

this->Controls->Add(this->incomePerMonthLabel);

this->Controls->Add(this->label16);

this->Controls->Add(this->calculateIncomeButton);

this->Controls->Add(this->countOfBoxesPerMonthTextBox);

this->Controls->Add(this->label12);

this->Controls->Add(this->boxPriceLabel);

this->Controls->Add(this->label14);

this->Controls->Add(this->calculateBoxPriceButton);

this->Controls->Add(this->boxCostLabel);

this->Controls->Add(this->label13);

this->Controls->Add(this->calculateBoxCostButton);

this->Controls->Add(this->label11);

this->Controls->Add(this->equipmentDeprecationPriceTextBox);

this->Controls->Add(this->costsLabel);

this->Controls->Add(this->label10);

this->Controls->Add(this->calculateCostsButton);

this->Controls->Add(this->label9);

this->Controls->Add(this->label8);

this->Controls->Add(this->label7);

this->Controls->Add(this->taxRateTextBox);

this->Controls->Add(this->salaryRateTextBox);

this->Controls->Add(this->utilityRateTextBox);

this->Controls->Add(this->materialPriceLabel);

this->Controls->Add(this->label6);

this->Controls->Add(this->calculateMaterialPriceButton);

this->Controls->Add(this->label5);

this->Controls->Add(this->label4);

this->Controls->Add(this->label3);

this->Controls->Add(this->drawBox);

this->Controls->Add(this->label2);

this->Controls->Add(this->label1);

this->Controls->Add(this->sortDescbutton);

this->Controls->Add(this->sortAscbutton);

this->Controls->Add(this->saveButton);

this->Controls->Add(this->materialsDataGridView);

this->Controls->Add(this->importMaterialsButton);

this->Controls->Add(this->addButton);

this->Controls->Add(this->priceTextBox);

this->Controls->Add(this->nameTextBox);

this->Controls->Add(this->drawButton);

this->Controls->Add(this->boxDepthTextBox);

this->Controls->Add(this->boxHeightTextBox);

this->Controls->Add(this->boxWidthTextBox);

this->FormBorderStyle = System::Windows::Forms::FormBorderStyle::FixedSingle;

this->Margin = System::Windows::Forms::Padding(4);

this->Name = L"MyForm";

this->Text = L"BoxCreator";

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->materialsDataGridView))->EndInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->drawBox))->EndInit();

this->ResumeLayout(false);

this->PerformLayout();

}

#pragma endregion

std::list<Material\*>\* materials = new std::list<Material\*>();

private: System::Void addButton\_Click(System::Object^ sender, System::EventArgs^ e) { // додає новий матеріал до таблиці

try {

auto price = System::Convert::ToDouble(priceTextBox->Text);

Material\* material = new Material(toStandardString(nameTextBox->Text), price);

int index = materialsDataGridView->Rows->Add();

materialsDataGridView->Rows[index]->Cells[0]->Value = gcnew String(material->getName().c\_str());

materialsDataGridView->Rows[index]->Cells[1]->Value = material->getPrice();

materials->push\_back(material);

}

catch (FormatException^ ex) {

MessageBox::Show(this, "Bad price/name format", "Validation error", MessageBoxButtons::OK); // виводить повідомлення, якщо прайс

// або імя були вказані неправильно

}

}

static std::string toStandardString(System::String^ string) { // конвертує String до string

using System::Runtime::InteropServices::Marshal;

System::IntPtr pointer = Marshal::StringToHGlobalAnsi(string);

char\* charPointer = reinterpret\_cast<char\*>(pointer.ToPointer());

std::string returnString(charPointer, string->Length);

Marshal::FreeHGlobal(pointer);

return returnString;

}

private: System::Void importMaterialsButton\_Click(System::Object^ sender, System::EventArgs^ e) { // зчитує матеріали з файул і записує в таблицю

std::ifstream file("output.txt");

if (file.is\_open()) {

materials = new std::list<Material\*>();

materialsDataGridView->Rows->Clear();

Material\* material = new Material();

while (file >> material) {

auto rowIndex = materialsDataGridView->Rows->Add();

materialsDataGridView->Rows[rowIndex]->Cells[0]->Value = gcnew String(material->getName().c\_str());

materialsDataGridView->Rows[rowIndex]->Cells[1]->Value = material->getPrice();

materials->push\_back(material);

material = new Material();

}

}

else {

MessageBox::Show(this, "Saved file doesn't exist", "Validation error", MessageBoxButtons::OK);

// виводить повідомлення, якщо файлу не існує

}

file.close();

}

private: System::Void saveButton\_Click(System::Object^ sender, System::EventArgs^ e) { // зберігає дані з таблиці у файл

std::ofstream file("output.txt", std::ios::trunc);

for (auto it = materials->begin(); it != materials->end(); ++it) {

file << \*it;

}

file.close();

}

private: System::Void sortAscbutton\_Click(System::Object^ sender, System::EventArgs^ e) { // сортує в зростаючому порядку

int size = materials->size();

double\* arrayToSort = new double[size];

std::string\* names = new std::string[size];

int counter = 0;

for (auto it = materials->begin(); it != materials->end(); ++it) {

auto material = \*it;

arrayToSort[counter] = material->getPrice();

names[counter] = material->getName();

counter++;

}

double min = 0;

int count = 0;

int mainIndex = 0;

double temp = 0;

std::string tempName;

std::string minByPrice;

while (mainIndex < size) {

min = arrayToSort[mainIndex];

count = mainIndex;

for (int i = mainIndex; i < size; i++) {

if (arrayToSort[i] < min) {

min = arrayToSort[i];

minByPrice = names[i];

count = i;

}

}

temp = arrayToSort[mainIndex];

tempName = names[mainIndex];

arrayToSort[mainIndex] = min;

names[mainIndex] = minByPrice;

arrayToSort[count] = temp;

names[count] = tempName;

mainIndex++;

}

materials->clear();

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

Material\* element = new Material(names[i], arrayToSort[i]);

materials->push\_back(element);

}

printMaterials();

}

inline void printMaterials() { // виводить на екран таблицю

materialsDataGridView->Rows->Clear();

int i = 0;

for (auto it = materials->begin(); it != materials->end(); ++it) {

auto material = \*it;

materialsDataGridView->Rows->Add();

materialsDataGridView->Rows[i]->Cells[0]->Value = gcnew String(material->getName().c\_str());

materialsDataGridView->Rows[i]->Cells[1]->Value = material->getPrice();

i++;

}

}

private: System::Void sortDescbutton\_Click(System::Object^ sender, System::EventArgs^ e) { // сортує в спадаючому порядку

int size = materials->size();

double\* arrayToSort = new double[size];

std::string\* names = new std::string[size];

int counter = 0;

for (auto it = materials->begin(); it != materials->end(); ++it) {

auto material = \*it;

arrayToSort[counter] = material->getPrice();

names[counter] = material->getName();

counter++;

}

double max = 0;

int count = 0;

int mainIndex = 0;

double temp = 0;

std::string tempName;

std::string minByPrice;

while (mainIndex < size) {

max = arrayToSort[mainIndex];

count = mainIndex;

for (int i = mainIndex; i < size; i++) {

if (arrayToSort[i] > max) {

max = arrayToSort[i];

minByPrice = names[i];

count = i;

}

}

temp = arrayToSort[mainIndex];

tempName = names[mainIndex];

arrayToSort[mainIndex] = max;

names[mainIndex] = minByPrice;

arrayToSort[count] = temp;

names[count] = tempName;

mainIndex++;

}

materials->clear();

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

Material\* element = new Material(names[i], arrayToSort[i]);

materials->push\_back(element);

}

printMaterials();

}

private: System::Void button1\_Click(System::Object^ sender, System::EventArgs^ e) { // перемальовує коробку

drawBox->Invalidate();

}

private: Box\* getBox() { // створює бокс з заданими параметрами

double width = System::Convert::ToDouble(boxWidthTextBox->Text);

double height = System::Convert::ToDouble(boxHeightTextBox->Text);

double depth = System::Convert::ToDouble(boxDepthTextBox->Text);

Box\* box = new Box(width, height, depth);

return box;

}

private: System::Void drawBox\_Paint(System::Object^ sender, System::Windows::Forms::PaintEventArgs^ e) { // малює розкрій коробки за

// допомогою даних про коробку

try {

auto box = getBox();

auto myCanvas = e->Graphics;

System::Drawing::Pen^ pen = gcnew System::Drawing::Pen(System::Drawing::Color::Black);

double smallSideWidth = (double)box->width / 5.2;

int heightStart = drawBox->Height / 2 - box->height / 2;

int x = 10;

myCanvas->DrawRectangle(pen, x, heightStart, smallSideWidth, box->depth);

x += smallSideWidth;

myCanvas->DrawRectangle(pen, x, heightStart, box->width, box->depth);

x += box->width;

myCanvas->DrawRectangle(pen, x, heightStart, box->height, box->depth);

myCanvas->DrawRectangle(pen, x, heightStart - box->height / 10.0, box->height, (double)box->height / 10.0);

myCanvas->DrawRectangle(pen, x, heightStart + box->depth, box->height, (double)box->height / 10.0);

x += box->height;

myCanvas->DrawRectangle(pen, x, heightStart, box->width, box->depth);

myCanvas->DrawRectangle(pen, x, heightStart - box->height, box->width, box->height);

myCanvas->DrawRectangle(pen, x, heightStart + box->depth, box->width, box->height);

x += box->width;

myCanvas->DrawRectangle(pen, x, heightStart, box->height, box->depth);

}

catch (Exception^ ex) { // якщо не задані параметри

ex->Message;

}

}

private: System::Void calculateMaterialPriceButton\_Click(System::Object^ sender, System::EventArgs^ e) {// підраховує та виводить на екран

// ціну за коробку

if (checkMaterialIsEmpty()) { // перевіряє чи матеріали пусті

return;

}

if (checkRowIsNotSelected()) { // перевіряє чи рядок не вибраний

return;

}

if (checkBoxInput()) { // перевіряє чи введені всі дані для коробкир

return;

}

auto box = getBox();

std::string name = toStandardString(materialsDataGridView->SelectedRows[0]->Cells[0]->Value->ToString());

auto material = findMaterial(name);

Order\* order = new Order(box, material);

materialPriceLabel->Text = order->calculateMaterialPrice().ToString("#,0.00");

}

private: Material\* findMaterial(std::string name) { // знаходить матеріал в списку

for (auto it = materials->begin(); it != materials->end(); ++it) {

auto material = \*it;

if (material->getName() == name) {

return material;

}

}

}

private: bool checkMaterialIsEmpty() { // перевіряє чи матеріали пусті

try {

if (!materials->empty()) {

return false;

}

else {

throw MaterialsAreEmptyException();

}

}

catch (MaterialsAreEmptyException) {

MessageBox::Show(this, "Materials are empty. Please add new material", "Validation error", MessageBoxButtons::OK);

// виводить повідомлення на екран, якщо матеріали пусті

return true;

}

}

private: bool checkRowIsNotSelected() { // перевіряє чи не обраний рядок

try {

if (materialsDataGridView->SelectedRows->Count != 0) {

return false;

}

else {

throw RowIsNotSelectedException();

}

}

catch (RowIsNotSelectedException) {

MessageBox::Show(this, "Need to select material", "Validation error", MessageBoxButtons::OK);

// виводить поідомлення на екран, якщо рядок не вибраний

return true;

}

}

private: bool checkCostsInvalid() { // перевіряє чи введені дані про комунальні послуги, зарплату та податки

try {

double utilityRate = System::Convert::ToDouble(utilityRateTextBox->Text);

double salaryRate = System::Convert::ToDouble(salaryRateTextBox->Text);

double taxRate = System::Convert::ToDouble(taxRateTextBox->Text);

}

catch (System::FormatException^ ex) {

MessageBox::Show(this, "Cost inputs are not number. Try to change utility rate/salary rate/tax rate", "Validation error", MessageBoxButtons::OK);

// виводить повідомлення, якщо дані були заповнені некоректно

return true;

}

return false;

}

private: bool checkEquipmentDeprecationPrice() { // перевіряє чи введені дані про амортизацію обладнання

try {

double equipmentDeprecationPrice = System::Convert::ToDouble(equipmentDeprecationPriceTextBox->Text);

}

catch (System::FormatException^ ex) {

MessageBox::Show(this, "Equipment Deprecation Price is not a number", "Validation error", MessageBoxButtons::OK);

// виводить повідомлення, якщо амортизація обладнання не була вказана або вказана неправильно

return true;

}

return false;

}

private: bool checkCountOfBoxes() { // перевіряє чи введені дані про кількість коробок

try {

int countOfBoxesPerMonth = System::Convert::ToInt32(countOfBoxesPerMonthTextBox->Text);

}

catch (System::FormatException^ ex) {

MessageBox::Show(this, "Count of Boxes is not a number", "Validation error", MessageBoxButtons::OK);

// виводить повідомлення, якщо неправильно введені дані про кількість коробок

return true;

}

return false;

}

private: bool checkBoxInput() { // перевіряє чи глибина, довжина та ширина коробки правильно введені

try {

double width = System::Convert::ToDouble(boxWidthTextBox->Text);

double height = System::Convert::ToDouble(boxHeightTextBox->Text);

double depth = System::Convert::ToDouble(boxDepthTextBox->Text);

}

catch (System::FormatException^ ex) {

MessageBox::Show(this, "Width/Height/Depth is not a number", "Validation error", MessageBoxButtons::OK);

// виводить повідомлення , якщо дані введені неправильно

return true;

}

return false;

}

private: System::Void calculateCostsButton\_Click(System::Object^ sender, System::EventArgs^ e) { // підраховує витрати

// та виводить на екран

if (checkMaterialIsEmpty()) // перевіряє чи матеріали пусті

return;

if (checkRowIsNotSelected()) // перевіряє чи рядок не вибраний

return;

if (checkBoxInput()) // перевіряє чи правильно введені дані коробки

return;

if (checkCostsInvalid()) { // перевіряє чи додаткові дані правильно введені

return;

}

auto box = getBox();

std::string name = toStandardString(materialsDataGridView->SelectedRows[0]->Cells[0]->Value->ToString());

auto material = findMaterial(name);

Order\* order = new Order(box, material);

double utilityRate = System::Convert::ToDouble(utilityRateTextBox->Text);

double salaryRate = System::Convert::ToDouble(salaryRateTextBox->Text);

double taxRate = System::Convert::ToDouble(taxRateTextBox->Text);

costsLabel->Text = order->calculateCosts(utilityRate, salaryRate, taxRate).ToString("#,0.00");

}

private: System::Void calculateBoxCostButton\_Click(System::Object^ sender, System::EventArgs^ e) { // підраховує ціну коробки

// та виводить на екран

if (checkMaterialIsEmpty()) // перевіряє чи матеріали пусті

return;

if (checkRowIsNotSelected()) // перевіряє чи рядок не вибраний

return;

if (checkBoxInput()) // перевіряє чи правильно введені дані коробки

return;

if (checkCostsInvalid()) { // перевіряє чи додаткові дані правильно введені

return;

}

if (checkEquipmentDeprecationPrice()) // перевіряє чи правильно введена амортизація обладання

return;

auto box = getBox();

std::string name = toStandardString(materialsDataGridView->SelectedRows[0]->Cells[0]->Value->ToString());

auto material = findMaterial(name);

Order\* order = new Order(box, material);

double utilityRate = System::Convert::ToDouble(utilityRateTextBox->Text);

double salaryRate = System::Convert::ToDouble(salaryRateTextBox->Text);

double taxRate = System::Convert::ToDouble(taxRateTextBox->Text);

double equipmentDeprecationPrice = System::Convert::ToDouble(equipmentDeprecationPriceTextBox->Text);

boxCostLabel->Text = order->calculateBoxCost(utilityRate, salaryRate, taxRate, equipmentDeprecationPrice).ToString("#,0.00");

}

private: System::Void calculateBoxPriceButton\_Click(System::Object^ sender, System::EventArgs^ e) { // підраховує ціну коробки для купівлі

// та виводить на екран

if (checkMaterialIsEmpty()) // перевіряє чи матеріали пусті

return;

if (checkRowIsNotSelected()) // перевіряє чи рядок не вибраний

return;

if (checkBoxInput()) // перевіряє чи правильно введені дані коробки

return;

if (checkCostsInvalid()) { // перевіряє чи додаткові дані правильно введені

return;

}

if (checkEquipmentDeprecationPrice()) // перевіряє чи правильно введена амортизація обладання

return;

auto box = getBox();

std::string name = toStandardString(materialsDataGridView->SelectedRows[0]->Cells[0]->Value->ToString());

auto material = findMaterial(name);

Order\* order = new Order(box, material);

double utilityRate = System::Convert::ToDouble(utilityRateTextBox->Text);

double salaryRate = System::Convert::ToDouble(salaryRateTextBox->Text);

double taxRate = System::Convert::ToDouble(taxRateTextBox->Text);

double equipmentDeprecationPrice = System::Convert::ToDouble(equipmentDeprecationPriceTextBox->Text);

boxPriceLabel->Text = order->calculateBoxPrice(utilityRate, salaryRate, taxRate, equipmentDeprecationPrice).ToString("#,0.00");

}

private: System::Void calculateIncomeButton\_Click(System::Object^ sender, System::EventArgs^ e) { // підраховує загальний дохід

// та вивдить на екран

if (checkMaterialIsEmpty()) // перевіряє чи матеріали пусті

return;

if (checkRowIsNotSelected()) // перевіряє чи рядок не вибраний

return;

if (checkBoxInput()) // перевіряє чи правильно введені дані коробки

return;

if (checkCostsInvalid()) { // перевіряє чи додаткові дані правильно введені

return;

}

if (checkEquipmentDeprecationPrice()) // перевіряє чи правильно введена амортизація обладання

return;

if (checkCountOfBoxes()) // перевіряє чи кількість коробок правильно введена

return;

auto box = getBox();

std::string name = toStandardString(materialsDataGridView->SelectedRows[0]->Cells[0]->Value->ToString());

auto material = findMaterial(name);

Order\* order = new Order(box, material);

double utilityRate = System::Convert::ToDouble(utilityRateTextBox->Text);

double salaryRate = System::Convert::ToDouble(salaryRateTextBox->Text);

double taxRate = System::Convert::ToDouble(taxRateTextBox->Text);

double equipmentDeprecationPrice = System::Convert::ToDouble(equipmentDeprecationPriceTextBox->Text);

int countOfBoxesPerMonth = System::Convert::ToInt32(countOfBoxesPerMonthTextBox->Text);

incomePerMonthLabel->Text = order->calculateIncomePerMonth(utilityRate, salaryRate, taxRate, equipmentDeprecationPrice, countOfBoxesPerMonth).ToString("#,0.00");

}

private: System::Void calculateAllbutton\_Click(System::Object^ sender, System::EventArgs^ e) { // підраховує всі функції

if (checkMaterialIsEmpty()) // перевіряє чи матеріали пусті

return;

if (checkRowIsNotSelected()) // перевіряє чи рядок не вибраний

return;

if (checkBoxInput()) // перевіряє чи правильно введені дані коробки

return;

if (checkCostsInvalid()) { // перевіряє чи додаткові дані правильно введені

return;

}

if (checkEquipmentDeprecationPrice()) // перевіряє чи правильно введена амортизація обладання

return;

if (checkCountOfBoxes()) // перевіряє чи кількість коробок правильно введена

return;

auto box = getBox();

std::string name = toStandardString(materialsDataGridView->SelectedRows[0]->Cells[0]->Value->ToString());

auto material = findMaterial(name);

Order\* order = new Order(box, material);

double utilityRate = System::Convert::ToDouble(utilityRateTextBox->Text);

double salaryRate = System::Convert::ToDouble(salaryRateTextBox->Text);

double taxRate = System::Convert::ToDouble(taxRateTextBox->Text);

double equipmentDeprecationPrice = System::Convert::ToDouble(equipmentDeprecationPriceTextBox->Text);

int countOfBoxesPerMonth = System::Convert::ToInt32(countOfBoxesPerMonthTextBox->Text);

materialPriceLabel->Text = order->calculateMaterialPrice().ToString("#,0.00");

costsLabel->Text = order->calculateCosts(utilityRate, salaryRate, taxRate).ToString("#,0.00");

boxCostLabel->Text = order->calculateBoxCost(utilityRate, salaryRate, taxRate, equipmentDeprecationPrice).ToString("#,0.00");

boxPriceLabel->Text = order->calculateBoxPrice(utilityRate, salaryRate, taxRate, equipmentDeprecationPrice).ToString("#,0.00");

incomePerMonthLabel->Text = order->calculateIncomePerMonth(utilityRate, salaryRate, taxRate, equipmentDeprecationPrice, countOfBoxesPerMonth).ToString("#,0.00");

}

};

}

Назва файлу: MyForm.cpp

#include "MyForm.h"

using namespace kursova;

int main() {

Application::EnableVisualStyles();

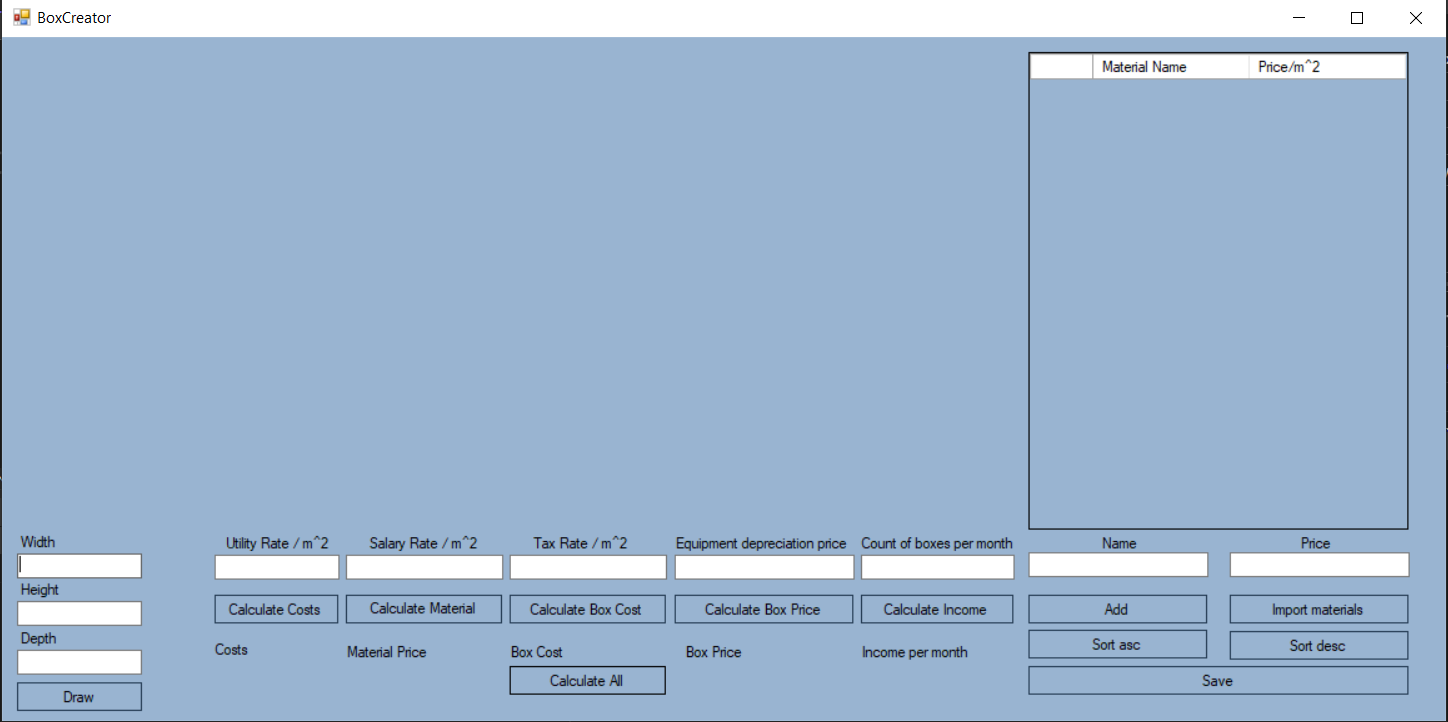
Application::SetCompatibleTextRenderingDefault(false);

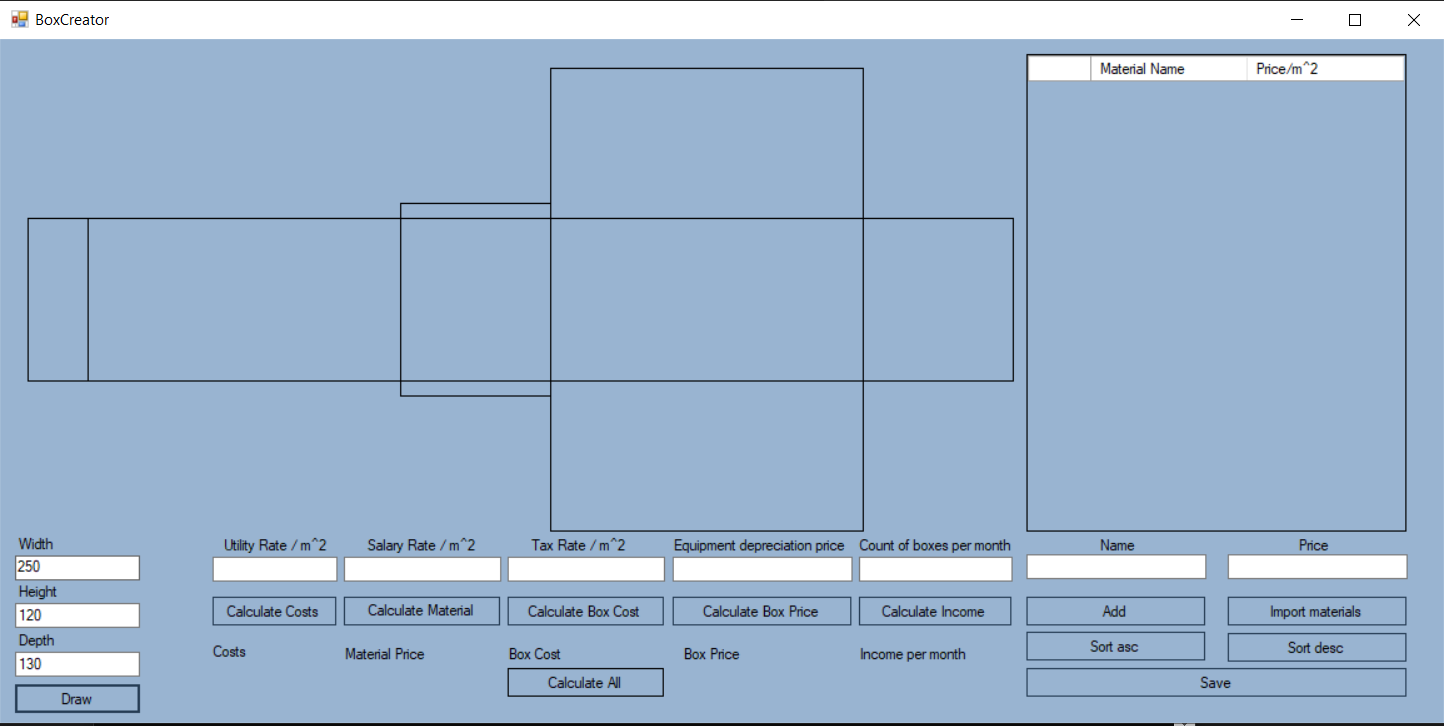
Application::Run(gcnew MyForm());

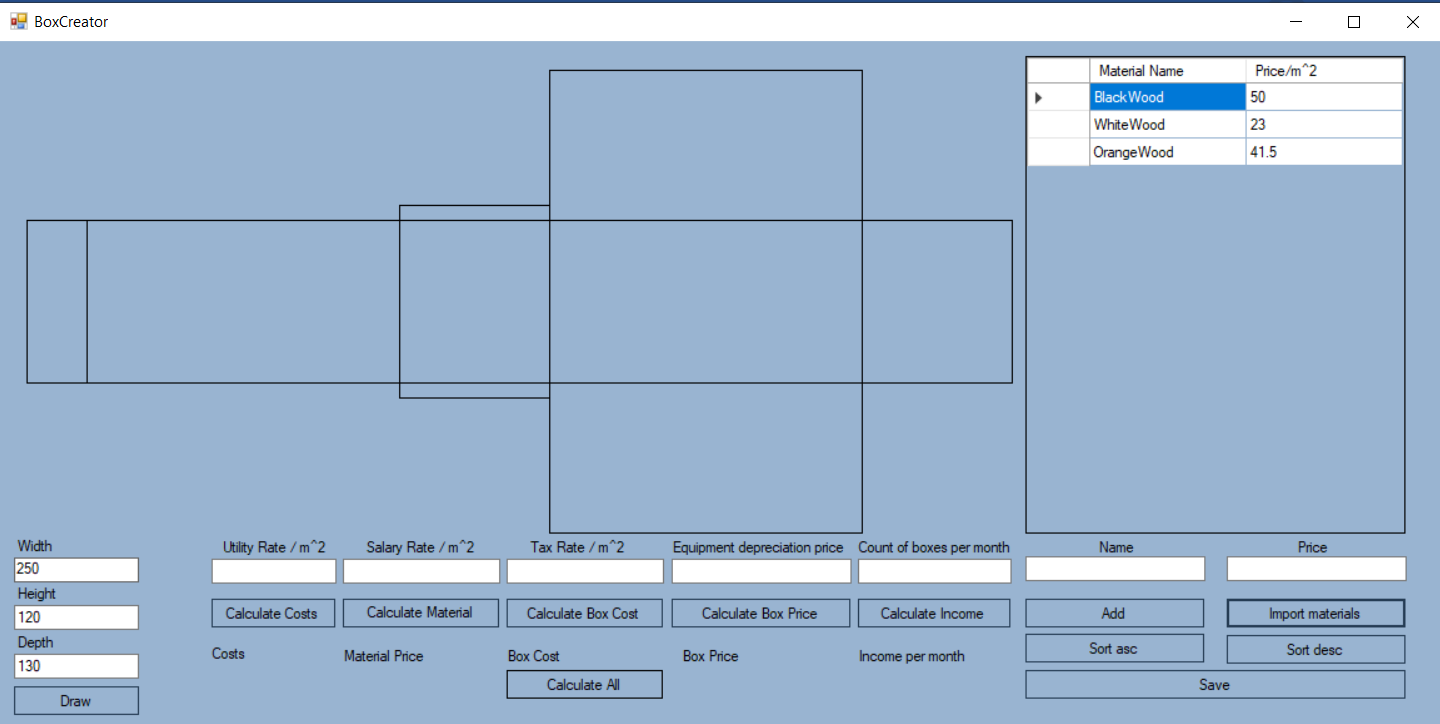
return 0;

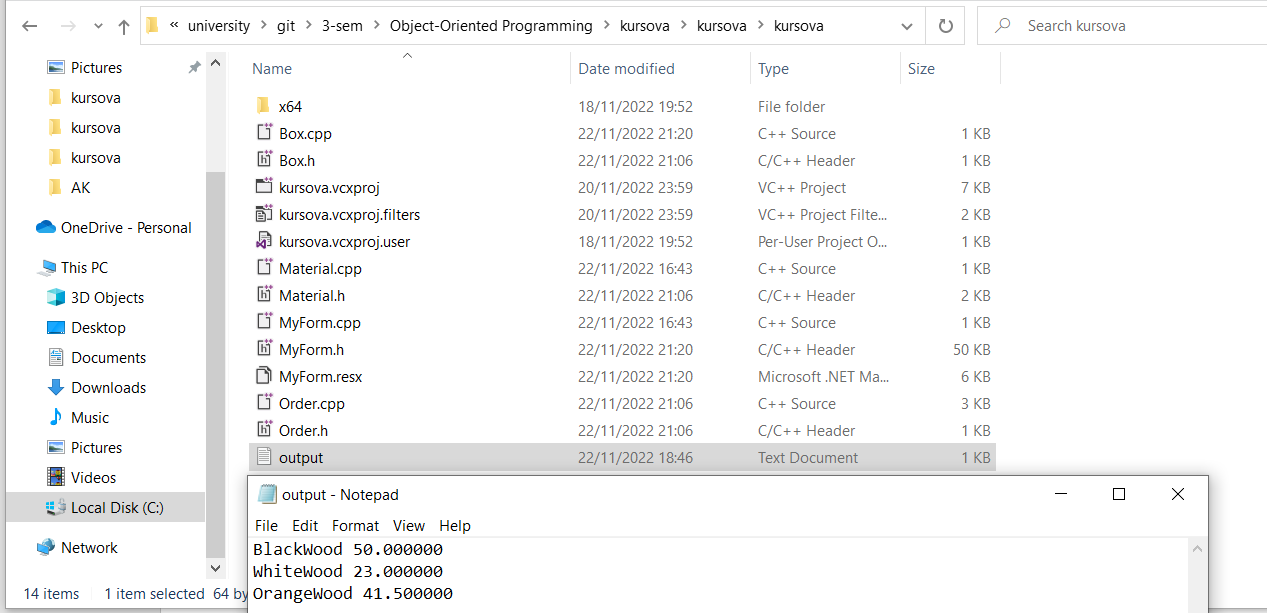
}

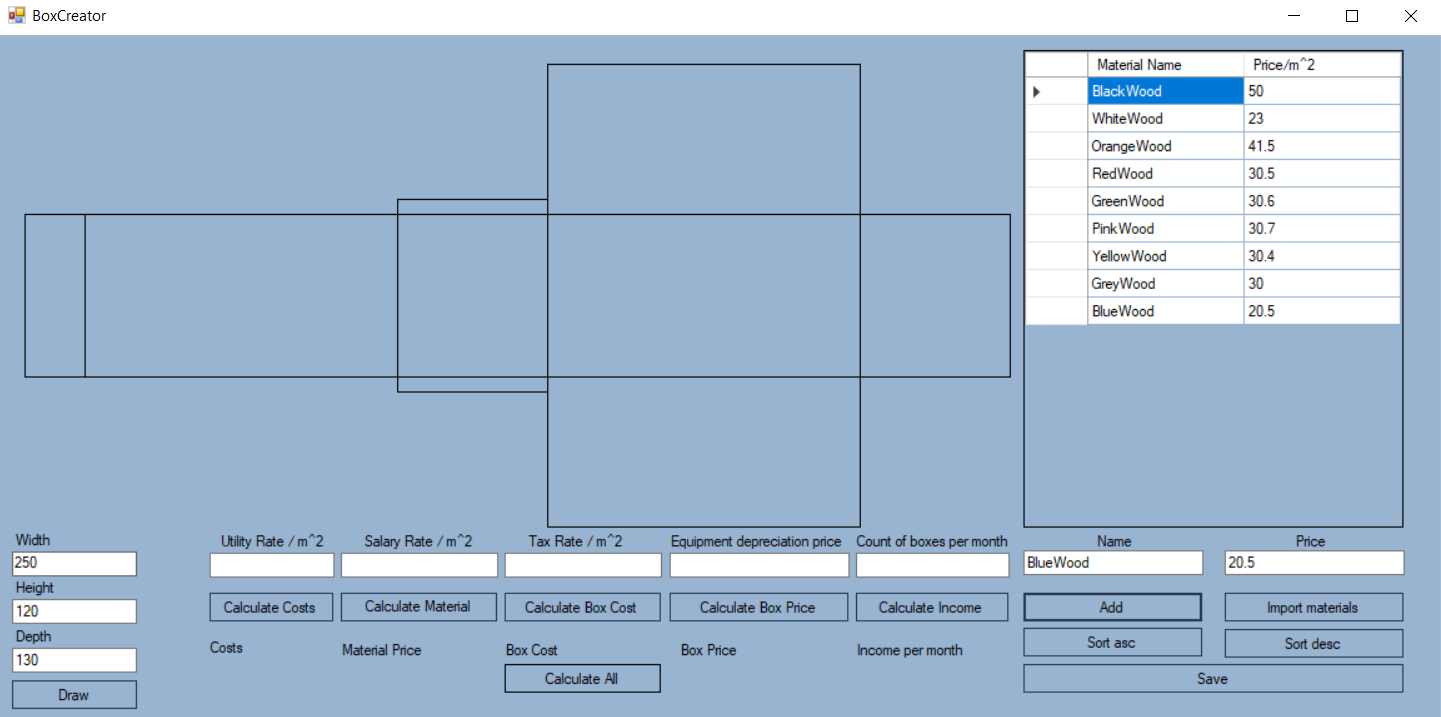
1. **Протокол роботи програми для кожного пункту завдання**

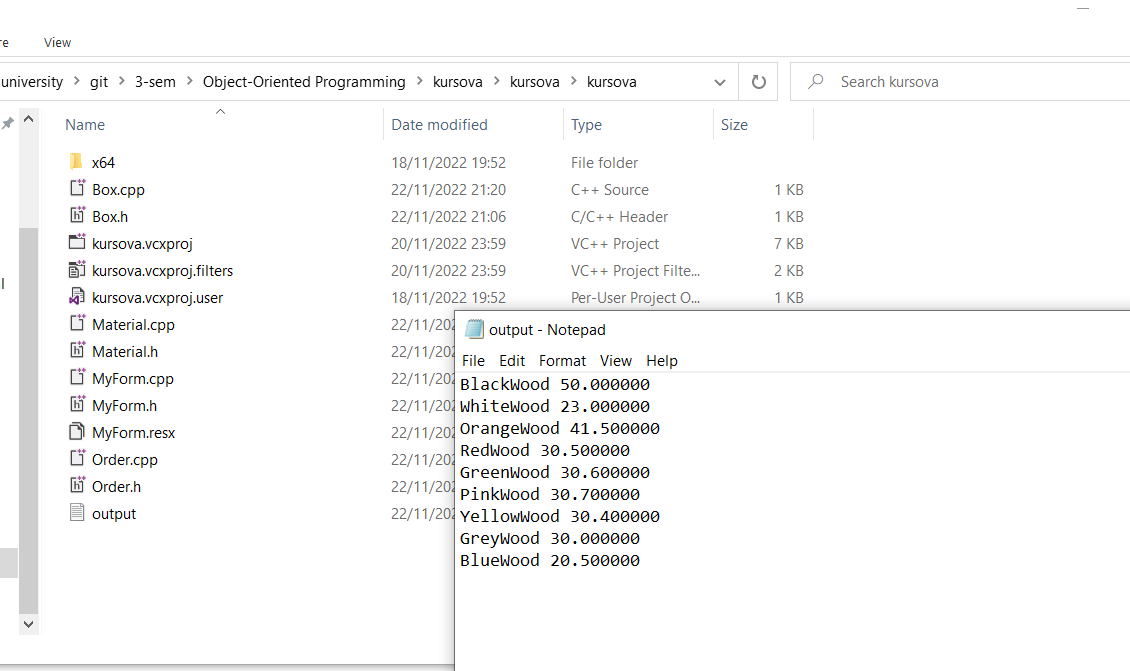
*****Рис. 4.1. Головне вікно програми.*

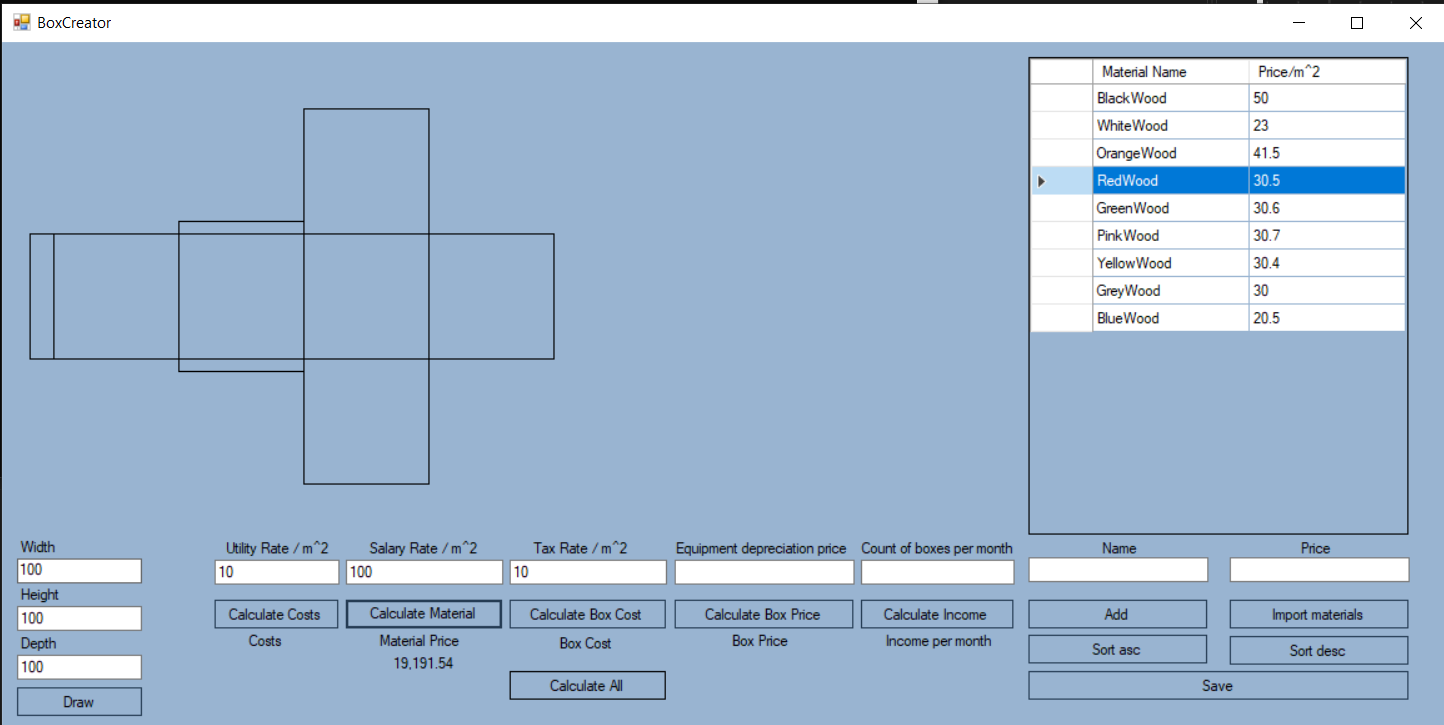
*****Рис. 4.2. Розкрій прямокутної коробки за заданими довжиною, шириною та глибиною.*

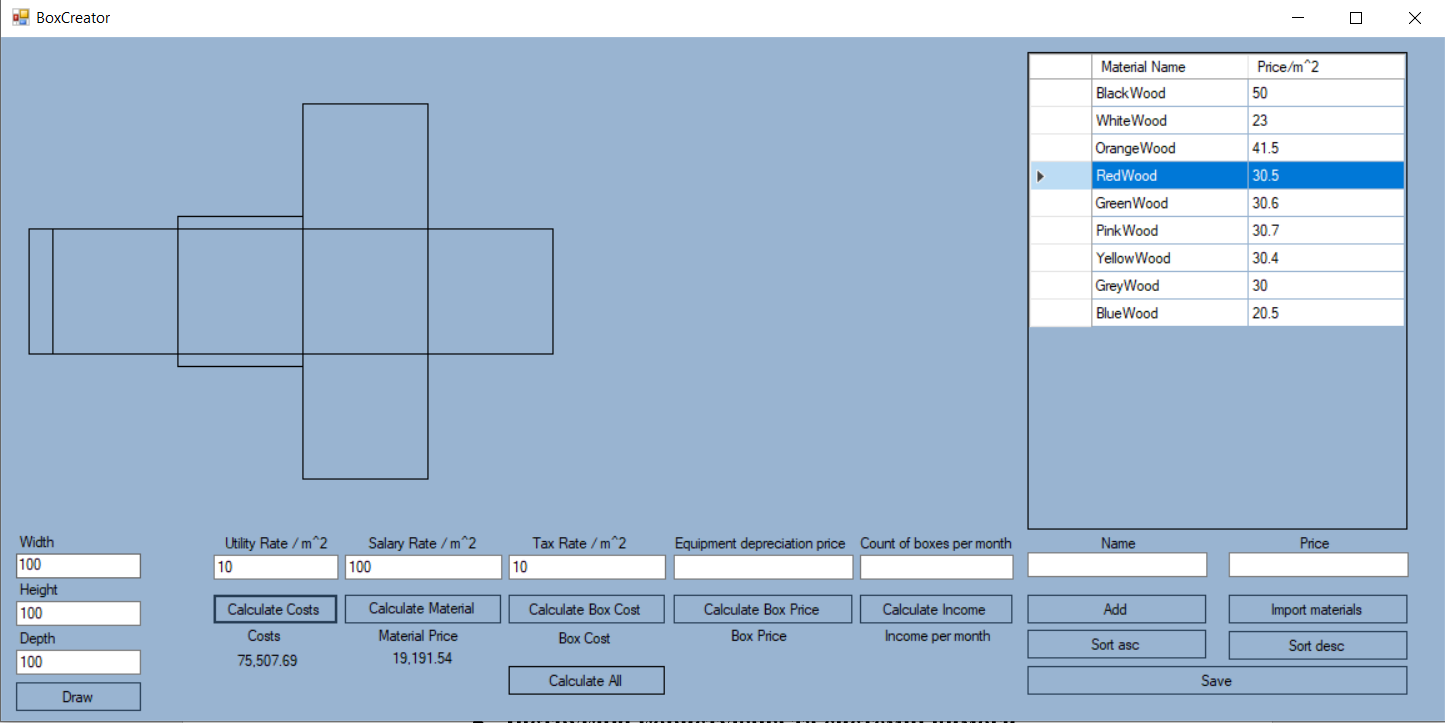
*****Рис. 4.3. Таблиця матеріалів та ціни за квадратний метр після натискання кнопки «Import materials», яка виконує операцію зчитування матеріалів з файлу.*

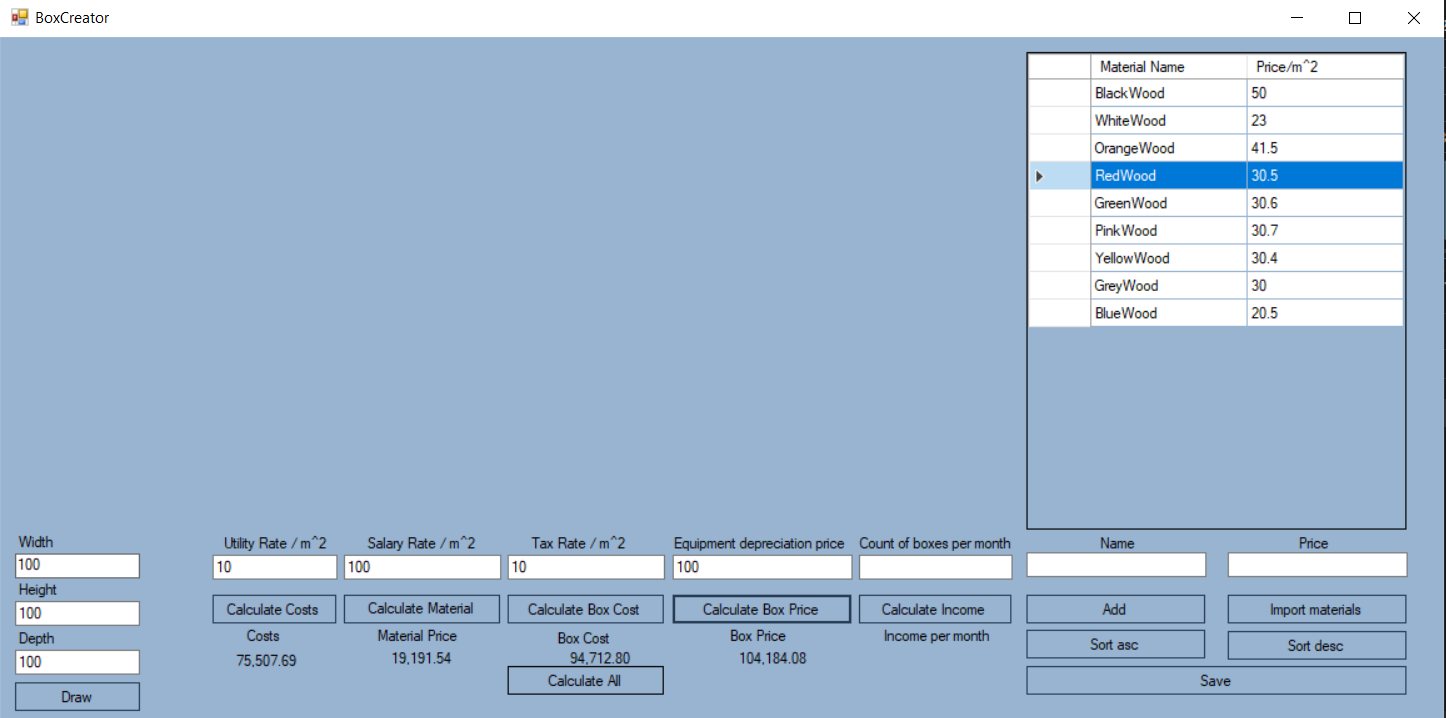
*****Рис. 4.4 Вікно вибору файлу, з якого зчитуються дані матеріалів у таблицю та вміст файлу.*

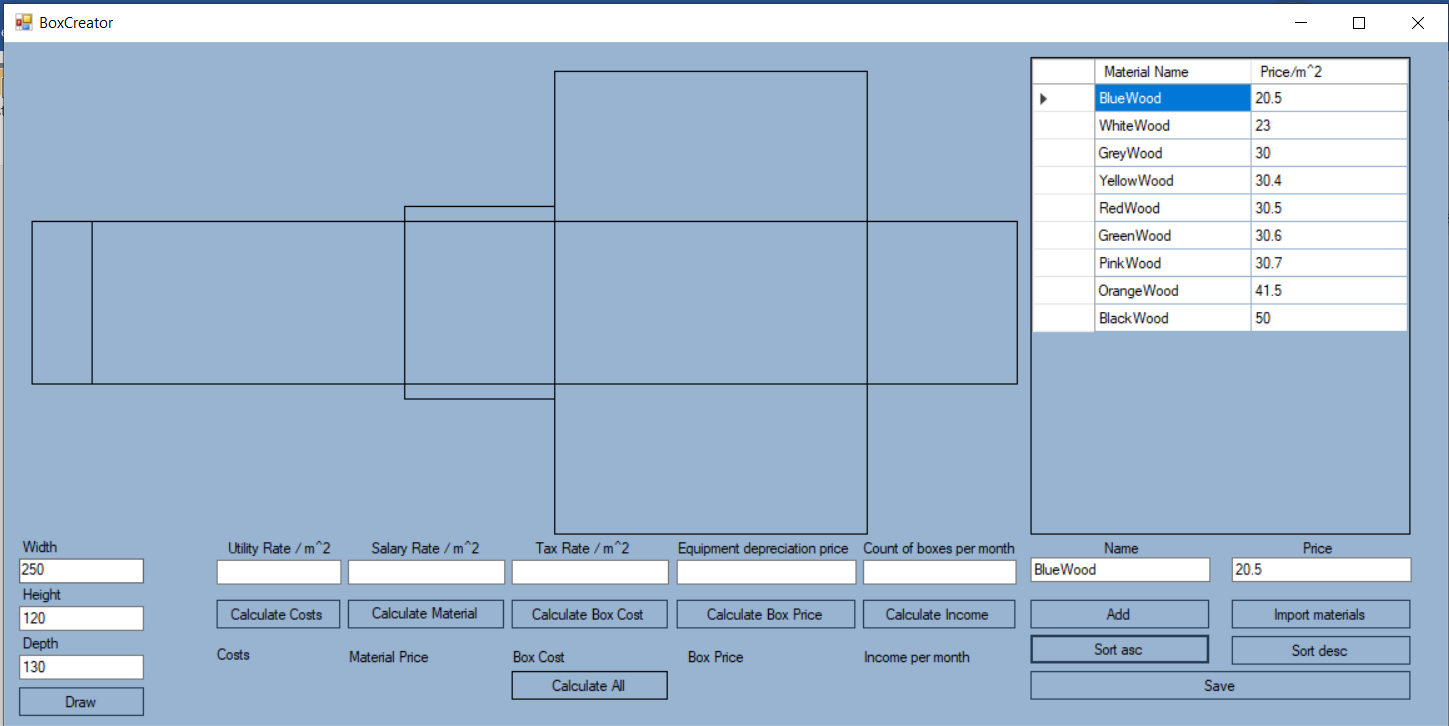
*****Рис. 4.5. Таблиця матеріалів та ціни за квадратний метр при додаванні матеріалів з клавіатури за допомогою кнопки «Add».*

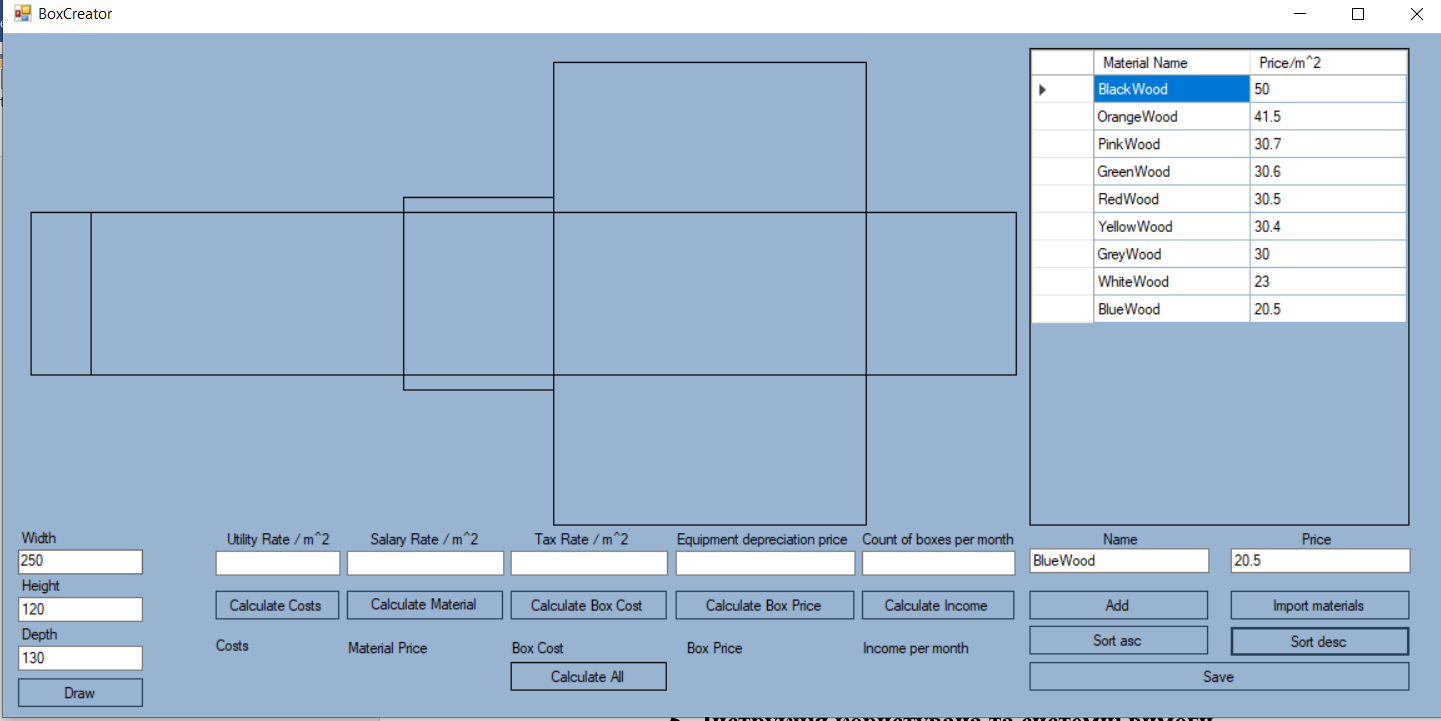
*****Рис. 4.6. Вікно вибору файлу та вміст файлу при натисканні кнопки «Save».*

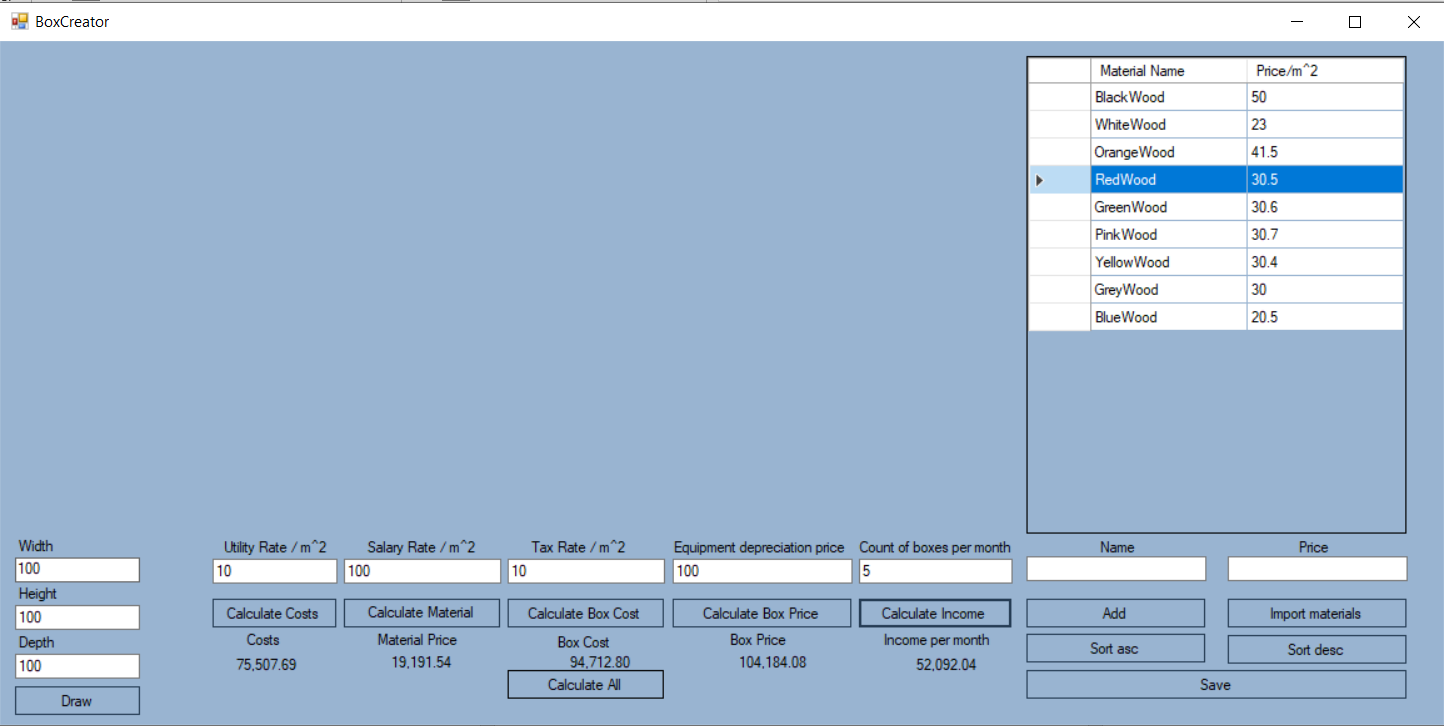
*****Рис. 4.7. Результат вирахування собівартості матеріалів за допомогою кнопки «Calculate Material».*

*****Рис. 4.8. Результат вирахування витрат, таких як комунальні послуги, податки та зарплата за допомогою кнопки «Calculate costs».*

*Рис. 4.9 Результат вирахування вартості виготовлення коробки, ціни купівлі за допомогою кнопок «Calculate Box Cost» «Calculate Box Price».*

*****Рис. 4.10. Таблиця матеріалів та ціни за квадратний метр , посортованої у зростаючому порядку за допомогою кнопки «Sort asc».*

*****Рис. 4.11. Таблиця матеріалів та ціни за квадратний метр , посортованої у спадаючому порядку за допомогою кнопки «Sort desc».*

*Рис. 4.12. Результат вирахування доходу за допомогою кнопки «Calculate Income».*

1. **Інструкція користувача та системні вимоги**
   1. **Компоненти ПЗ**

Програмне забезпечення розроблено на мові програмуванні C++ у середовищі розробки Visual Studio 2022 з використанням Windows Forms, яке може експлуатуватися під управлінням сімейства операційних систем Windows. Під час програмування було використано об’єктно-орієнтований підхід. Для коректної роботи програмного забезпечення необхідно мати користувацьку машину з процесором , у якого тактова частота не менше 400 MHz та операційна пам’ять не менше 2 ГБ.

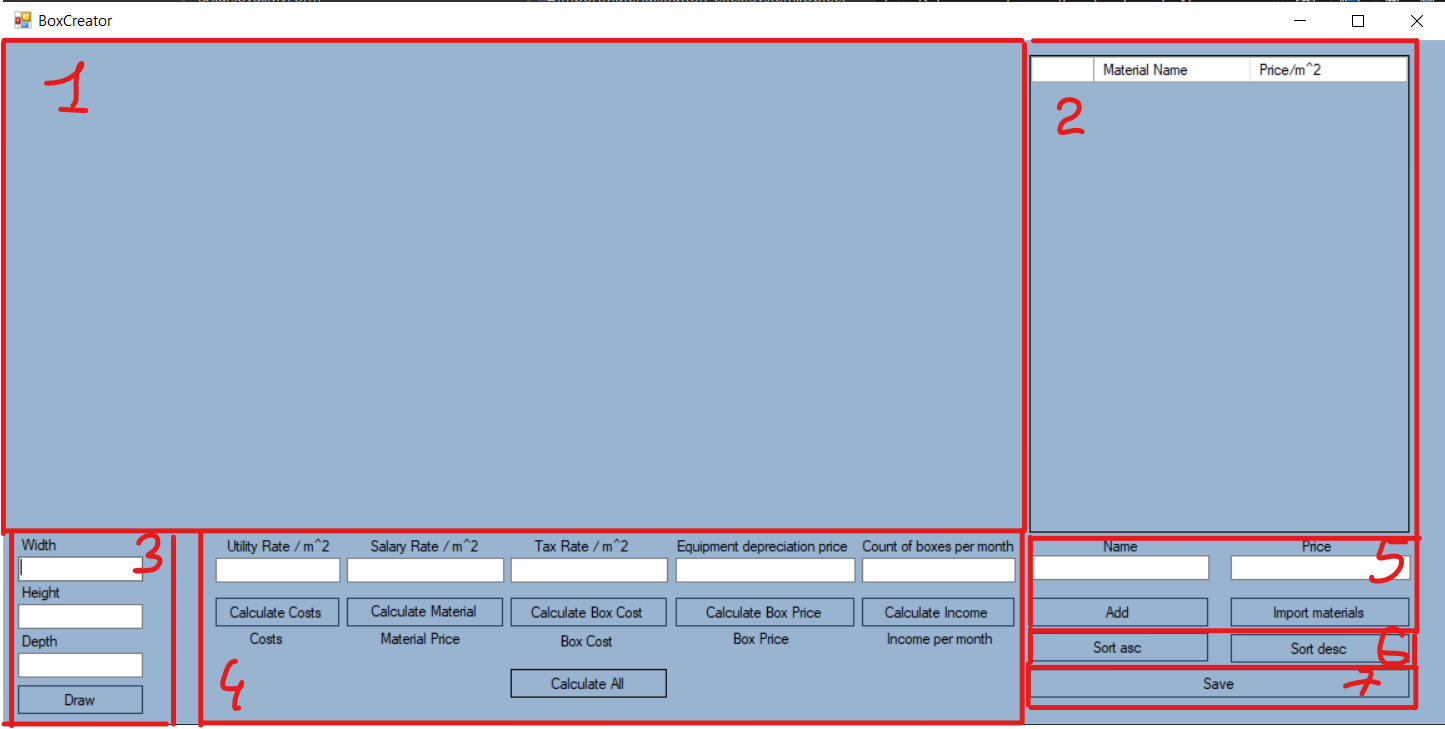
Для коректної роботи програми необхідно мати файл kursova.exe та усі необхідні бібліотеки та текстовий файл із заповненими даними.

* 1. **Встановлення ПЗ**

1. Потрібно встановити середовище Visual Studio 2022 або попередніх років.
2. Потрібно встановити пакет kursova, щоб отримати доступ до файлу kursova.exe
3. Додати до пакету текстовий файл із назвами матеріалв та цінами за квадратний метр.
   1. **Налаштування ПЗ**

Потрібно запустити файл на виконання у Visual Studio 2022.

* 1. **Базові функції ПЗ**

*****Рис. 5.1. Головне вікно програми, поділене на секції відповідно до базових функцій ПЗ.*

**Секція 1:**

Спеціально визначене місце, де буде малюватись розкрій прямокутної коробки за параметрами: довжина, ширина і глибина. Щоб редагувати це місце, використовується секція 3.

**Секція 2:**

Таблиця матеріалів та відповідних їм цін за квадратний метр. Щоб редагувати таблицю, використовуються секції 5, 6, 7.

**Секція 3:**

Три поля, які відповідають за ширину , довжину та глибину коробки. Щоб отримати коректний результат у секції 1, потрібно ввести в цих трьох полях додатні дійсні числа, при яких ширина не перевищуватиме 240, довжина і глибина не перевищуватимуть 125. Потрібно натиснути кнопку **«Draw»,** що розташована нижче трьох полей вводу, і розкрій з’явиться у секції 1.

**Секція 4:**

Ця секція відповідає за розрахунок витрат, доходу , ціни матеріалу, собівартості коробки та ціни коробки. Для того, щоб отримати результати, потрібно виконати **усі умови із секції 3,** а також **завантажити дані** про матеріали та ціни за метр квадратний і **обрати** відповідний рядок з даними, про що детальніше у секції 5.

Щоб отримати усі результати, потрібно заповнити поля **«Utility Rate / m^2», «Salary Rate / m^2», «Tax Rate / m^2», «Equipment depreciation price», «Count of boxes per month»** додатними дійсними числами і натиснути кнопку **«Calculate all».**

Також можна вирахувати все окремо:

Щоб отримати розрахунок витрат, потрібно заповнити поля **«Utility Rate / m^2», «Salary Rate / m^2», «Tax Rate / m^2»** додатними дійсними числами і натиснути кнопку **«Calculate costs».**

Щоб отримати розрахунок вартості матеріалів, потрібно лише натиснути кнопку **«Calculate Material».**

Щоб отримати розрахунок собівартості коробки, потрібно заповнити поля **«Utility Rate / m^2», «Salary Rate / m^2», «Tax Rate / m^2» , «Equipment depreciation price»** додатними дійсними числами та натиснути кнопку **«Calculate Box Cost».**

Щоб отримати розрахунок кінцевої вартості коробки, потрібно заповнити поля **«Utility Rate / m^2», «Salary Rate / m^2», «Tax Rate / m^2» , «Equipment depreciation price»** додатними дійсними числами та натиснути кнопку **«Calculate Box Price».**

Щоб отримати розрахунок доходу, потрібно заповнити поля **«Utility Rate / m^2», «Salary Rate / m^2», «Tax Rate / m^2», «Equipment depreciation price», «Count of boxes per month»** додатними дійсними числами і натиснути кнопку «**Calculate Income».**

**Секція 5:**

Ця секція відповідає за додавання матеріалу та цін за квадратний метр у таблицю, передбачену секцією 2. Є два варіанти додавання: з файлу або ж додавання вручну з клавіатури.

Щоб додати дані з файлу, потрібно дотриматись усіх умов, пов’язаних із файлом, про що сказано у розділі 7, а також натиснути кнопку **«Import Materials»**.

Щоб додати дані з клавіатури, потрібно заповнити поле **«Name»**, у якому повинно міститись одне слово. Якщо ж назва матеріалу передбачає два слова і більше, потрібно записати усі слова без пробілів , кожне з великої літери,тим самим утворити одне слово. А також потрібно заповнити поле **«Price»**, де має ввестись додатнє дійсне число. Після цього, потрібно натиснути кнопку **«Add»** та дані з’являться в таблиці.

**Секція 6:**

Тут з’являється можливість посортувати дані, які були попередньо завантажені , дотримуючись умов із секції 5, з таблиці, область якої передбачена секцією 2.

Для того, щоб посортувати дані за ціною за квадратний метр у зростаючому порядку, потрібно натиснути кнопку **«Sort asc»**.

Для того, щоб посортувати дані за ціною за квадратний метр у спадаючому порядку, потрібно натиснути кнопку **«Sort desc»**.

Секція 7:

Ця секція надає можливість зберігання даних із таблиці у файл, потрібно лише натиснути кнопку **«Save».**

* 1. **Аналіз можливих помилок**

При помилці, коли не малюється розкрій прямокутної коробки, потрібно перевірити чи довжина, ширина та глибина були введені за типом даних – додатнє дійсне число.

При помилці некоректного зчитування даних, потрібно перевірити чи неушкоджений файл, також чи він знаходиться у пакеті , чи правильно введене ім’я файлу і чи виконуються усі умови із розділу 7.

При усіх інших помилках детальніше розписано у розділі 6.

* 1. **Системні вимоги**

Процесор: 32-розрядний або 64-розрядний процесор із тактовою частотою 1 ГГц або швидший (наприклад, IntelCore i3);

ОЗП: 2 ГБ ОЗП (в т.ч. 50 МБ вільних, решта - для підтримки операційної системи) або більше.

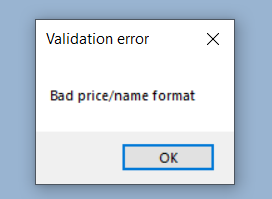
Інше обладнання: під’єднана клавіатура, або інший метод стандартного вводу, наявність монітора або іншого методу стандартного виводу.

Графічний пристрій із підтримкою DirectX 9 і драйвером WDDM 1.0 або новішим.

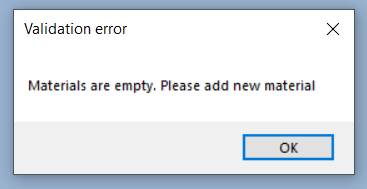
Рекомендована ОС: Windows 7 або вище/MacOS 10.14 або вище/аналогічні вищезгаданим ОС на ядрі Linux. 6

Спеціальне програмне забезпечення: не вимагається.

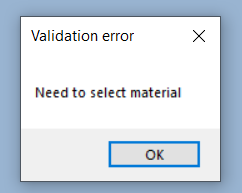
1. **Опис виняткових ситуацій**

****

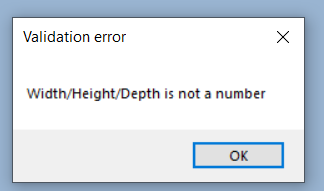
*Рис. 6.1. Повідомлення про помилку при неправильному вводі назви матеріалу або ціни матеріалу за квадратний метр та натисканні кнопки «Add».*

****

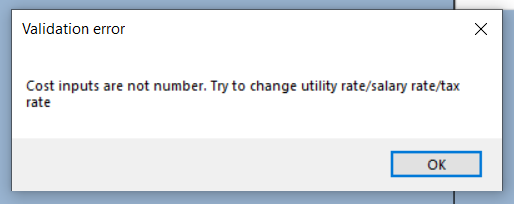
*Рис. 6.2. Повідомлення про помилку, якщо користувач не додав матеріалів до таблиці і натиснув кнопки «Calculate costs», «Calculate Material», «Calculate Box Cost», «Calculate Box Price», «Calculate Income», «Calculate all», які вираховують витрати, вартість матеріалу, вартість коробки, ціну коробки та дохід або ж вираховують усі пункти відповідно.*

**

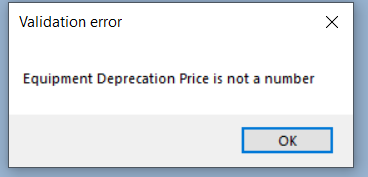
*Рис. 6.3. Повідомлення про помилку, якщо користувач додав матеріали до таблиці , але не обрав потрібний йому рядок, і натиснув кнопки «Calculate costs», «Calculate Material», «Calculate Box Cost», «Calculate Box Price», «Calculate Income», «Calculate all», які вираховують витрати, вартість матеріалу, вартість коробки, ціну коробки та дохід або ж вираховують усі пункти відповідно.*

**

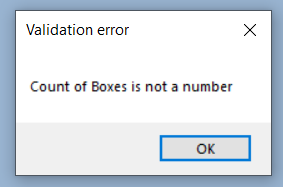
*Рис. 6.4. Повідомлення про помилку, якщо ширина, довжина або глибина коробки були введені неправильно або не введені взагалі і користувач натиснув кнопки «Calculate costs», «Calculate Material», «Calculate Box Cost», «Calculate Box Price», «Calculate Income», «Calculate all», які вираховують витрати, вартість матеріалу, вартість коробки, ціну коробки та дохід або ж вираховують усі пункти відповідно.*

**

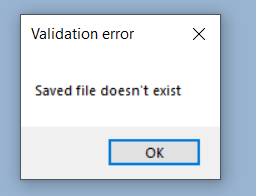
*Рис. 6.5. Повідомлення про помилку, якщо користувач не заповнив або заповнив неправильно поля, що відподають за ціну комунальних послуг, податків та зарплати за місяць та натиснув кнопки «Calculate costs», «Calculate Box Cost», «Calculate Box Price», «Calculate Income», «Calculate all», які вираховують витрати, вартість коробки, ціну коробки та дохід або ж вираховують усі пункти відповідно.*

**

*Рис. 6.6. Повідомлення про помилку, якщо поле, яке відповідає за ціну амортизації обладнання, було заповнене неправильно або не заповнене взагалі і користувач натиснув кнопки «Calculate Box Cost», «Calculate Box Price», «Calculate Income», «Calculate all», які вираховують вартість коробки, ціну коробки та дохід або ж вираховують усі пункти відповідно.*

**

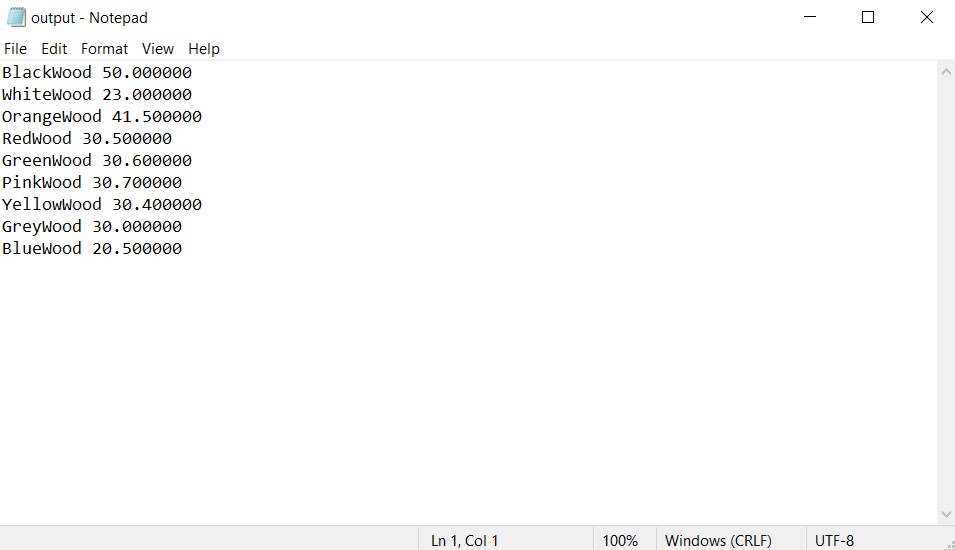
*Рис. 6.7. Повідомлення про помилку, якщо поле, яке відповідає за кількість коробок на місяць, було заповнене неправильно або не заповнене взагалі і користувач натиснув кнопки «Calculate Income», «Calculate all», які вираховують дохід або ж вираховують усі пункти відповідно*

**

*Рис. 6.8. Повідомлення про помилку, якщо збережений файл, з якого користувач намагається зчитати матеріали та ціни за квадратний метр, не існує при натисканні кнопки «Import Materials».*

1. **Структура файлу вхідних даних**

Дані повинні бути записані у текстовий файл з розширенням “.txt”. Приклад структури файлу вхідних даних зображено на рисунку:

*****Рис. 7.1. Коректний файл матеріалів та ціни за квадратний метр.*

Файл повинен містити пари матеріалу та ціни за квадратний метр, кожна з яких записується у новий рядок.

Спочатку записується назва матеріалу, яка має бути одним словом, у разі назви, більшої за одне слово потрібно записати кожне слово з великої літери без пробілів між ними, а тоді через пробіл записується ціна за квадратний метр – дійсне додатне число.

**Висновки**

**Список використаної літератури**