**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ**

**ФГБОУ ВО «Пензенский государственный университет»**

#### Кафедра «Математическое обеспечение и применение ЭВМ»

**ОТЧЕТ**

по лабораторной работе № 3

по дисциплине «Экономика программной инженерии»

на тему «Ценообразование на программные продукты»

Вариант 9

Выполнили:

Левин М.В.

Мастеров А.О.

Малышев А.С.

Принял: доцент Князев В.Н.

Пенза 2023

Цель работы

Приобретение практических навыков автоматизированных расчетов затрат (себестоимости), цены, нижнего предела цены и договорной цены на программные продукты (ПП).

Задание 1

Разработать на согласованном с преподавателем языке программу расчета затрат (себестоимости), цены, нижнего предела цены и договорной цены программного продукта. Осуществить тестирование разработанной программы с использованием данных из описанного примера, после чего решить с помощью, разработанной и протестированной программы задачу ценообразования в соответствии с заданным преподавателем вариантом задания.

Программа должна иметь графический интерфейс с полями для исходных данных и результатов с соответствующей экономической терминологией, причем с целью электронного документирования процесс решения задачи ценообразования должен протоколироваться и фиксироваться (исходные данные и результаты) самой программой в соответствующем файле.

Программа должна также представить полученные результаты в графическом виде:

* круговая диаграмма для затрат (себестоимости) на разработку ПП с указанием значений и долей (процентов) всех пяти составляющих этих затрат;
* диаграмма (гистограмма) для основных результатов решения задачи: затрат (себестоимости), цены, нижнего предела цены и договорной цены программного продукта.

Примечания.

1. Отсутствующие в таблице вариантов заданий числовые исходные данные следует взять из рассмотренного примера расчета.

2. Интерфейс программы должен предусматривать ее универсальный диалоговый характер (возможность ввода различных числовых значений исходных данных).

Таблица 1 – Варианты заданий

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **№ вар.**  **исх. дан.** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
| **Тразр,** дней | 100 | 120 | 130 | 140 | 150 | 100 | 120 | 130 | 140 | 150 |
| **Отр,** руб/день | 500 | 600 | 700 | 500 | 600 | 700 | 500 | 600 | 700 | 500 |
| **Кд** | 0,08 | 0,09 | 1,00 | 1,01 | 1,02 | 1,04 | 1,06 | 1,08 | 1,10 | 1,12 |
| **П,** % | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 |
| **Мп,** кВт | 0,3 | 0,4 | 0,5 | 0,6 | 0,5 | 0,4 | 0,3 | 0,4 | 0,5 | 0,6 |
| **Цэ,** руб**.** | 1,4 | 1,5 | 1,6 | 1,7 | 1,8 | 1,9 | 1,4 | 1,5 | 1,6 | 1,7 |
| **ЗПо,** руб/месяц | 11000 | 12000 | 13000 | 11000 | 12000 | 13000 | 11000 | 12000 | 13000 | 11000 |
| **Эо ,**штук | 1 | 2 | 1 | 2 | 1 | 2 | 1 | 2 | 1 | 2 |
| **С,** руб. | 20000 | 19000 | 18000 | 21000 | 22000 | 17000 | 16000 | 20000 | 19000 | 18000 |
| **Ткод,** дней | 45 | 55 | 65 | 65 | 70 | 45 | 55 | 65 | 65 | 70 |
| **Тотл,** дней | 30 | 40 | 45 | 50 | 55 | 30 | 40 | 45 | 50 | 55 |
| **Р,** % | 20 | 21 | 22 | 23 | 24 | 25 | 20 | 21 | 22 | 23 |
| **Н,** % | 13 | 14 | 15 | 13 | 14 | 15 | 13 | 14 | 15 | 13 |
| **N,** экземпляров | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 |
| **Пд,** % | 11 | 12 | 13 | 14 | 15 | 11 | 12 | 13 | 14 | 15 |

Ход работы

Была разработана программа на языке C#, которая позволяет рассчитать затрат (себестоимости), цены, нижнего предела цены и договорной цены программного продукта (листинг 1)

Листинг 1 – код программы

using System.Windows.Forms;

using System.Text.RegularExpressions;

using System.Collections.Generic;

using System.Text;

using System.Windows.Forms.DataVisualization.Charting;

using System.IO;

using System.Linq;

namespace epi\_3laba

{

    public partial class Form1 : Form

    {

        private const int WORKTIME\_COMPUTER\_PER\_YEAR\_IN\_HOURS = 8 \* 288;

        private const int NORM\_AMORTIZATION\_PERCENT           = 50;

        private const string POSITIVE\_DOUBLE\_PATTERN = @"^\d+((,|\.)\d{1,3})?$";

        private const string POSITIVE\_INT\_PATTERN    = @"^\d+$";

        private static readonly Dictionary<int, string> namesInput = new Dictionary<int, string>()

        {

            {0, "длительность разработки"},

            {1, "з/п разработчика"},

            {2, "коофициент доп. з/п"},

            {3, "процент накладных расходов"},

            {4, "потребляемая мощность ЭВМ"},

            {5, "цена 1 кВт-ч электроэнергии"},

            {6, "заработная плата в месяц персонала"},

            {7, "кол-во обслуживаемых компьютеров"},

            {8, "балансовая стоимость компьютера"},

            {9, "длительность кодирования"},

            {10, "длительность отладки"},

            {11, "норматив рентабельности"},

            {12, "НДС"},

            {13, "тиражирование"},

            {14, "дополнительная прибыль (%)"},

        };

        private InputParametrs inputParametrs;

        private OutputParametrs outputParametrs;

        public Form1()

        {

            InitializeComponent();

        }

        private void btnSolve\_Click(object sender, System.EventArgs e)

        {

            if (Solve())

            {

                DrawCharts();

            }

        }

        private void CheckWordsByPattern(StringBuilder sb, List<(int index, string value)> items, string pattern)

        {

            foreach (var (index, value) in items)

            {

                if (string.IsNullOrEmpty(value))

                {

                    sb.AppendLine($"Ничего не введено: '{namesInput[index]}'");

                }

                else if (!Regex.IsMatch(value, pattern))

                {

                    sb.AppendLine($"Ошибка ввода: '{namesInput[index]}'");

                }

            }

        }

        private bool Solve()

        {

            (int index, string value) totalDevelopDaysStr = (0, tbTotalDevelopDays.Text.Trim().Replace('.', ','));

            (int index, string value) salaryPerDayStr = (1, tbSalaryPerDay.Text.Trim().Replace('.', ','));

            (int index, string value) coefficientAdditionalSalaryStr = (2, tbCoefficientAdditionalSalary.Text.Trim().Replace('.', ','));

            (int index, string value) overheadPercentStr = (3, tbOverheadPercent.Text.Trim().Replace('.', ','));

            (int index, string value) powerStr = (4, tbPower.Text.Trim().Replace('.', ','));

            (int index, string value) priceElectricityPerHourStr = (5, tbPriceElectricityPerHour.Text.Trim().Replace('.', ','));

            (int index, string value) computerManagerSalaryStr = (6, tbComputerManagerSalary.Text.Trim().Replace('.', ','));

            (int index, string value) countManagedComputersStr = (7, tbCountManagedComputers.Text.Trim().Replace('.', ','));

            (int index, string value) balanceComputerPriceStr = (8, tbBalanceComputerPrice.Text.Trim().Replace('.', ','));

            (int index, string value) codingDaysStr = (9, tbCodingDays.Text.Trim().Replace('.', ','));

            (int index, string value) debugDaysStr = (10, tbDebugDays.Text.Trim().Replace('.', ','));

            (int index, string value) profitRateStr = (11, tbProfitRate.Text.Trim().Replace('.', ','));

            (int index, string value) VATStr = (12, tbVAT.Text.Trim().Replace('.', ','));

            (int index, string value) сountReplicStr = (13, tbCountReplic.Text.Trim().Replace('.', ','));

            (int index, string value) additionalProfitPercentStr = (14, tbAdditionalProfitPercent.Text.Trim().Replace('.', ','));

            var listOfIntedgers = new List<(int index, string value)>()

            {

                totalDevelopDaysStr,

                countManagedComputersStr,

                codingDaysStr,

                debugDaysStr,

                сountReplicStr

            };

            var listOfDoubles = new List<(int index, string value)>()

            {

                salaryPerDayStr,

                coefficientAdditionalSalaryStr,

                overheadPercentStr,

                powerStr,

                priceElectricityPerHourStr,

                computerManagerSalaryStr,

                balanceComputerPriceStr,

                profitRateStr,

                VATStr,

                additionalProfitPercentStr

            };

            var sb = new StringBuilder();

            CheckWordsByPattern(sb, listOfIntedgers, POSITIVE\_INT\_PATTERN);

            CheckWordsByPattern(sb, listOfDoubles, POSITIVE\_DOUBLE\_PATTERN);

            if (sb.Length > 0)

            {

                MessageBox.Show(sb.ToString(), "Ошибка ввода", MessageBoxButtons.OK, MessageBoxIcon.Warning);

                return false;

            }

            inputParametrs = new InputParametrs()

            {

                TotalDevelopDays = int.Parse(totalDevelopDaysStr.value),

                SalaryPerDay = double.Parse(salaryPerDayStr.value),

                CoefficientAdditionalSalary = double.Parse(coefficientAdditionalSalaryStr.value),

                OverheadPercent = double.Parse(overheadPercentStr.value),

                Power = double.Parse(powerStr.value),

                PriceElectricityPerHour = double.Parse(priceElectricityPerHourStr.value),

                ComputerManagerSalary = double.Parse(computerManagerSalaryStr.value),

                CountManagedComputers = int.Parse(countManagedComputersStr.value),

                BalanceComputerPrice = double.Parse(balanceComputerPriceStr.value),

                CodingDays = int.Parse(codingDaysStr.value),

                DebugDays = int.Parse(debugDaysStr.value),

                ProfitRate = double.Parse(profitRateStr.value),

                VAT = double.Parse(VATStr.value),

                CountReplic = int.Parse(сountReplicStr.value),

                AdditionalProfitPercent = double.Parse(additionalProfitPercentStr.value)

            };

            outputParametrs = Calculate(inputParametrs);

            tbCostsDevelopProject.Text = outputParametrs.CostsDevelopProject.ToString("0.00");

            tbDeveloperSalary.Text = outputParametrs.DeveloperSalary.ToString("0.00");

            tbAdditionalSalary.Text = outputParametrs.AdditionalSalary.ToString("0.00");

            tbSocialContributions.Text = outputParametrs.SocialContributions.ToString("0.00");

            tbOverheads.Text = outputParametrs.Overheads.ToString("0.00");

            tbOperatingCosts.Text = outputParametrs.OperatingCosts.ToString("0.00");

            tbProjectPrice.Text = outputParametrs.ProjectPrice.ToString("0.00");

            tbLowerPriceLimit.Text = outputParametrs.LowerPriceLimit.ToString("0.00");

            tbContractPrice.Text = outputParametrs.ContractPrice.ToString("0.00");

            return true;

        }

        private void DrawCharts()

        {

            pieChart.Legends.Clear();

            pieChart.Series.Clear();

            pieChart.Titles.Clear();

            pieChart.Titles.Add("Затраты на разработку ПП");

            string[] pieLegends =

            {

                "з/п разработчиков",

                "дополнительная з/п",

                "отчисления на соц.нужды",

                "накладные расходы",

                "эксплуатационные расходы"

            };

            foreach(var legend in pieLegends)

            {

                pieChart.Legends.Add(new Legend(legend) { Docking = Docking.Bottom });

            }

            double[] pieValues =

            {

                outputParametrs.DeveloperSalary,

                outputParametrs.AdditionalSalary,

                outputParametrs.SocialContributions,

                outputParametrs.Overheads,

                outputParametrs.OperatingCosts

            };

            var pieSeries = new Series("Pie series") { ChartType = SeriesChartType.Pie };

            pieSeries.Points.DataBindXY(pieLegends, pieValues);

            pieChart.Series.Add(pieSeries);

            foreach(var point in pieSeries.Points)

            {

                point.Label = point.YValues[0].ToString("0.00") + "\n" + (point.YValues[0] / outputParametrs.CostsDevelopProject).ToString("P");

                point.LegendText = point.AxisLabel;

            }

            /\* histogram \*/

            histogram.Legends.Clear();

            histogram.Series.Clear();

            histogram.Titles.Clear();

            histogram.Titles.Add("Затраты на разработку ПП");

            histogram.ChartAreas[0].RecalculateAxesScale();

            string[] histLegends =

            {

                "затраты на разработку",

                "цена разработанного ПО",

                "нижний предел цены",

                "договорная цена"

            };

            double[] histValues =

            {

                outputParametrs.CostsDevelopProject,

                outputParametrs.ProjectPrice,

                outputParametrs.LowerPriceLimit,

                outputParametrs.ContractPrice

            };

            var histSeries = new Series("Histogram series") { ChartType = SeriesChartType.Column };

            histSeries.Points.DataBindXY(histLegends, histValues);

            histogram.Series.Add(histSeries);

            foreach (var point in histSeries.Points)

            {

                point.Label = point.YValues[0].ToString("0.00");

            }

        }

        private OutputParametrs Calculate(InputParametrs inputParametrs)

        {

            var outputParametrs = new OutputParametrs();

            outputParametrs.DeveloperSalary = inputParametrs.TotalDevelopDays \* inputParametrs.SalaryPerDay;

            outputParametrs.AdditionalSalary = inputParametrs.CoefficientAdditionalSalary \* outputParametrs.DeveloperSalary;

            outputParametrs.SocialContributions = (outputParametrs.DeveloperSalary + outputParametrs.AdditionalSalary) \* 0.356;

            outputParametrs.Overheads = outputParametrs.DeveloperSalary \* inputParametrs.OverheadPercent / 100;

            double electricityCostsPerYear = inputParametrs.Power \* WORKTIME\_COMPUTER\_PER\_YEAR\_IN\_HOURS \* inputParametrs.PriceElectricityPerHour;

            double computerManageCostsPerYear = inputParametrs.ComputerManagerSalary \* 12 / inputParametrs.CountManagedComputers;

            double amortization = NORM\_AMORTIZATION\_PERCENT \* inputParametrs.BalanceComputerPrice / 100;

            double repairCosts = 0.05 \* inputParametrs.BalanceComputerPrice;

            double operatingCostsPerHour = (electricityCostsPerYear + computerManageCostsPerYear + amortization + repairCosts) / WORKTIME\_COMPUTER\_PER\_YEAR\_IN\_HOURS;

            outputParametrs.OperatingCosts = (inputParametrs.CodingDays + inputParametrs.DebugDays) \* 8 \* operatingCostsPerHour;

            outputParametrs.CostsDevelopProject = outputParametrs.DeveloperSalary + outputParametrs.AdditionalSalary + outputParametrs.SocialContributions + outputParametrs.Overheads + outputParametrs.OperatingCosts;

            outputParametrs.ProjectPrice = outputParametrs.CostsDevelopProject \* (1 + inputParametrs.ProfitRate / 100);

            outputParametrs.LowerPriceLimit = outputParametrs.ProjectPrice \* (1 + inputParametrs.VAT / 100) / inputParametrs.CountReplic;

            outputParametrs.ContractPrice = outputParametrs.LowerPriceLimit \* (1 + inputParametrs.AdditionalProfitPercent / 100);

            return outputParametrs;

        }

        private void saveToolStripMenuItem\_Click(object sender, System.EventArgs e)

        {

            var saveFileDialog = new SaveFileDialog()

            {

                Filter = "Epi file (\*.epi)|\*.epi",

                ValidateNames = false,

                CheckFileExists = true,

                CheckPathExists = true,

            };

            var dialogResult = saveFileDialog.ShowDialog();

            if (dialogResult == DialogResult.Cancel || dialogResult == DialogResult.Abort)

            {

                return;

            }

            using (StreamWriter writer = new StreamWriter(saveFileDialog.FileName, false))

            {

                writer.Write(inputParametrs.ToString());

                writer.Write($"{new string('-', 30)}\n");

                writer.Write(outputParametrs.ToString());

            }

        }

        private void loadToolStripMenuItem\_Click(object sender, System.EventArgs e)

        {

            var openFileDialog = new OpenFileDialog();

            var dialogResult = openFileDialog.ShowDialog();

            if (dialogResult == DialogResult.Cancel || dialogResult == DialogResult.Abort)

            {

                return;

            }

            using (StreamReader reader = new StreamReader(openFileDialog.FileName))

            {

                tbTotalDevelopDays.Text = reader.ReadLine().Split(' ').Reverse().Skip(1).First();

                tbSalaryPerDay.Text = reader.ReadLine().Split(' ').Reverse().Skip(1).First();

                tbCoefficientAdditionalSalary.Text = reader.ReadLine().Split(' ').Last();

                tbOverheadPercent.Text = reader.ReadLine().Split(' ').Reverse().Skip(1).First();

                tbPower.Text = reader.ReadLine().Split(' ').Reverse().Skip(1).First();

                tbPriceElectricityPerHour.Text = reader.ReadLine().Split(' ').Reverse().Skip(1).First();

                tbComputerManagerSalary.Text = reader.ReadLine().Split(' ').Reverse().Skip(1).First();

                tbCountManagedComputers.Text = reader.ReadLine().Split(' ').Reverse().Skip(1).First();

                tbBalanceComputerPrice.Text = reader.ReadLine().Split(' ').Reverse().Skip(1).First();

                tbCodingDays.Text = reader.ReadLine().Split(' ').Reverse().Skip(1).First();

                tbDebugDays.Text = reader.ReadLine().Split(' ').Reverse().Skip(1).First();

                tbProfitRate.Text = reader.ReadLine().Split(' ').Reverse().Skip(1).First();

                tbVAT.Text = reader.ReadLine().Split(' ').Reverse().Skip(1).First();

                tbCountReplic.Text = reader.ReadLine().Split(' ').Reverse().Skip(1).First();

                tbAdditionalProfitPercent.Text = reader.ReadLine().Split(' ').Reverse().Skip(1).First();

            }

            if (Solve())

            {

                DrawCharts();

            }

        }

        private void clearResultToolStripMenuItem\_Click(object sender, System.EventArgs e)

        {

            pieChart.Legends.Clear();

            pieChart.Series.Clear();

            pieChart.Titles.Clear();

            histogram.Legends.Clear();

            histogram.Series.Clear();

            histogram.Titles.Clear();

            tbTotalDevelopDays.Text = "";

            tbSalaryPerDay.Text = "";

            tbCoefficientAdditionalSalary.Text = "";

            tbOverheadPercent.Text = "";

            tbPower.Text = "";

            tbPriceElectricityPerHour.Text = "";

            tbComputerManagerSalary.Text = "";

            tbCountManagedComputers.Text = "";

            tbBalanceComputerPrice.Text = "";

            tbCodingDays.Text = "";

            tbDebugDays.Text = "";

            tbProfitRate.Text = "";

            tbVAT.Text = "";

            tbCountReplic.Text = "";

            tbAdditionalProfitPercent.Text = "";

            tbCostsDevelopProject.Text = "";

            tbDeveloperSalary.Text = "";

            tbAdditionalSalary.Text = "";

            tbSocialContributions.Text = "";

            tbOverheads.Text = "";

            tbOperatingCosts.Text = "";

            tbProjectPrice.Text = "";

            tbLowerPriceLimit.Text = "";

            tbContractPrice.Text = "";

        }

    }

}

Результат работы программы для данных из примера лабораторной работы представлен на рисунке 1.

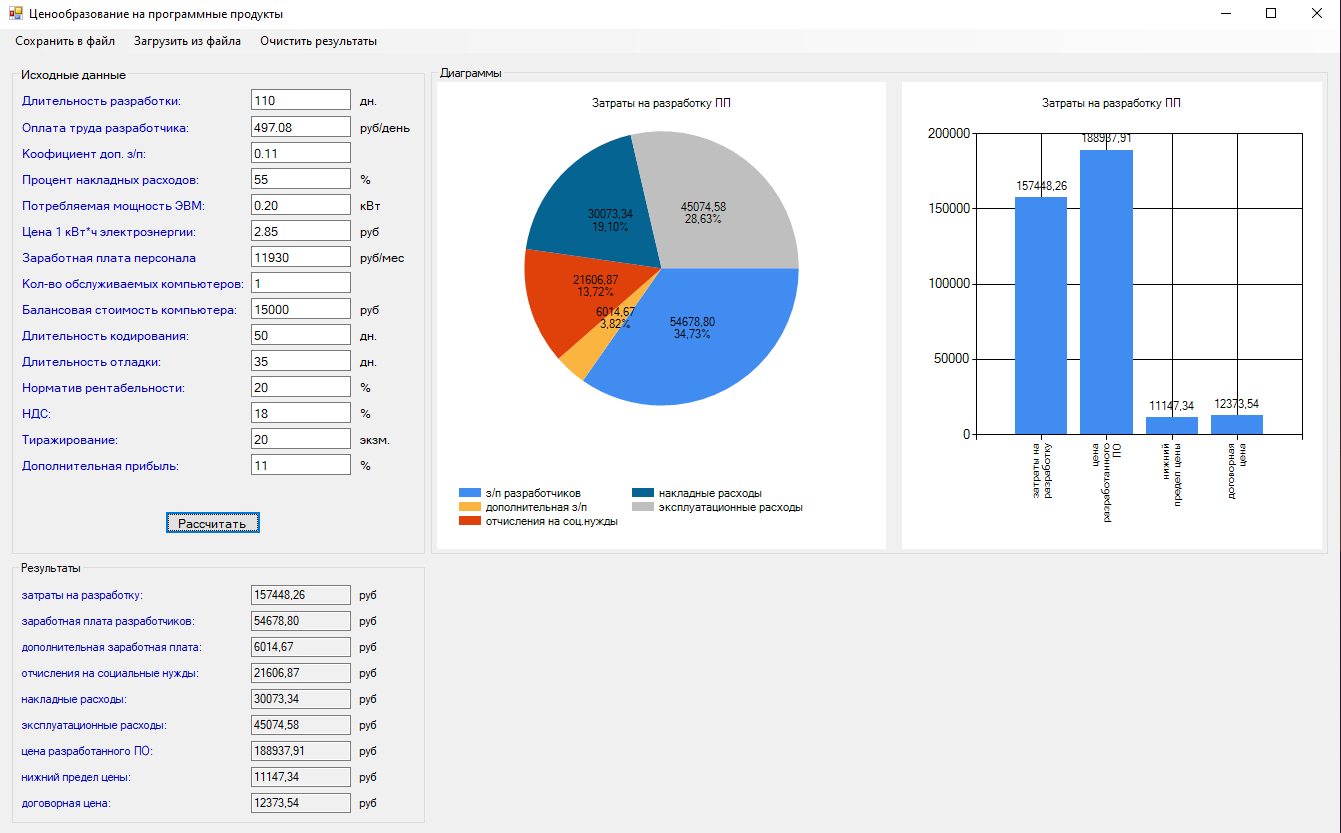


Рисунок 1 – Результат работы программы (данные из примера)

Результат работы программы для данных из 9 варианта таблицы 1 представлен на рисунке 2.

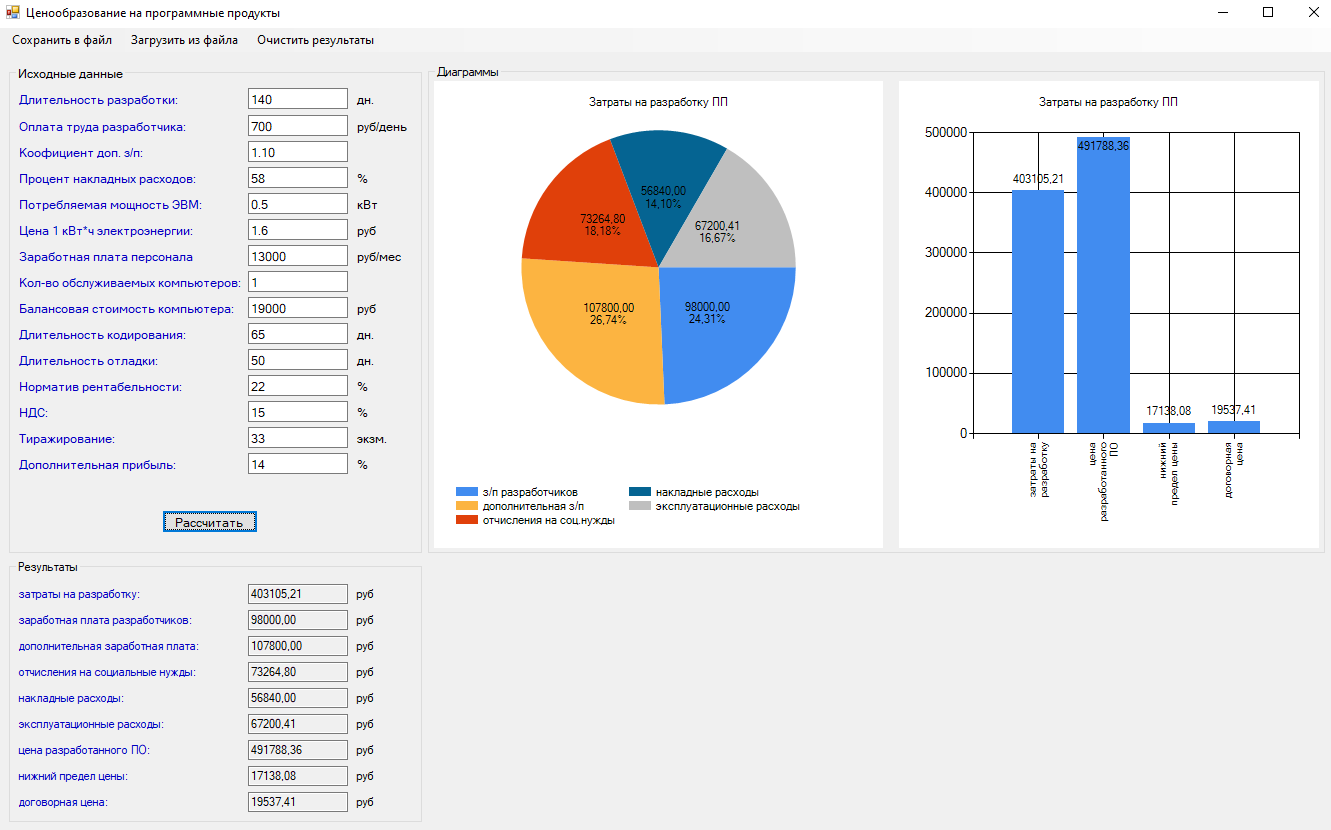


Рисунок 2 – Результат работы программы (8 вариант)

**Вывод:**

Приобрели практических навыки автоматизированных расчетов затрат (себестоимости), цены, нижнего предела цены и договорной цены на программные продукты (ПП).

Была разработана программа на языке C#, которая позволяет рассчитать затрат (себестоимости), цены, нижнего предела цены и договорной цены программного продукты.

При помощи программы были получены следующие результаты:

затраты: 403105,21 руб.

з/п разработчиков: 98000,00 руб.

дополнительная з/п: 107800,00 руб.

отчисления на соц. нужды: 73264,80 руб.

накладные расходы: 56840,00 руб.

эксплуатационные расходы: 67200,41 руб.

цена: 491788,36 руб.

нижний предел цены: 17138,08 руб.

договорная цена: 19537,41 руб.