ГУАП

КАФЕДРА № 43

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ст. преподаватель |  |  |  | Н.А. Соловьева |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №6 |
| «КАЧЕСТВО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ. ОЦЕНКА КАЧЕСТВА ПРОГРАММНОГО ПРОДУКТА.» |
| по курсу: ТЕХНОЛОГИЯ ПРОГРАММИРОВАНИЯ |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 4941 |  |  |  | Н.С. Горбунов |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2021

**1. Цель работы:**

Целью данной работы является изучение методологии оценки качества программного продукта на основе одной из существующих методик.

**2. Задание на лабораторную работу**

Оценить качество программного продукта по перечисленным в методике параметрам.

**3. Оценки качества программного продукта по всем указанным в методичке параметрам**

Для оценки была использована программа, написанная для дисциплины «Операционные системы» мои коллегой, а именно «Эмулятор диспетчера задач».

**3.1 Показатели качества**

**3.1.1 Показатели «завершенность»**

* Средства для очистки ОЗУ реализованы на уровне «Garbage collector» языка C#.
* Окно программы можно изменять в размерах. Однако адаптивная верстка позволяет сохранять все элементы управления в поле видимости пользователя.
* В программе присутствует поле для ввода целого числа-константы. При вводе некорректных значений программа прекращает свою работу.
* Данные, использующиеся в программе, заносятся через специальный «тест-файл». Файл помещается в поле для ввода входных данных. При некорректных данных выводится ошибка. Программа продолжает работать в штатном режиме
* Все циклы и переменные прошли автоматические тесты, ошибок не выявлено, значения в пределах нормы.
* Программа зависит только от стандартных библиотек .NET Framework, следовательно, будет корректно работать на всех устройствах системы Windows.

**3.1.2 Показатели «стандартизация»**

* Все константы кода определены стандартно и уникально.
* Структура арифметический выражений соответствует одной принятой на проекте методологии.
* Все переменные имеют уникальные имена, согласно их смысловому значению. Все наименования даны согласно принятой на проекте методологии.

**3.1.3 Показатели «рациональность»**

* Оптимизация часто используемых фрагментов кода присутствует в достаточном объеме. Оптимизация была произведена с помощью повышения уровня абстракции, согласно парадигме ООП

**3.1.4 Показатели «доступность»**

* Числа, подверженные изменениям в программе не использованы.

**3.1.5 Показатели «коммуникативность»**

* Программа адаптирована под гибкий объем входных данных.
* Сообщения об ошибках выводятся согласно классификации языка. В основном дают сообщения для разработчика.

**3.1.6 Показатели «структурированность»**

* В каждой части подпрограммы имеется количество точек выхода не менее одной.
* Приложение имеет одну глобальную точку выхода. Локальные ошибки модулей к терминальной ошибке не приводят. Исключением является ввод некорректной константы.
* Подпрограммы соответствуют собственному функциональному назначению.

**3.1.7 Показатели «Информативность»**

* Каждый модуль имеет собственный комментарий в принятом проектной методологией формате.
* Для описания зависимостей модулей предназначены отдельные блоки комментариев.
* Имена объектов соответствуют собсвенному назначению.

**3.1.8 Показатели «Осмысленность»**

* Все операторы выполнимы при тестировании.
* Вычисления, не относящиеся к циклу, выполняются вне его структуры.
* Для предоставления значений «истина» или «ложь» в коде использованы логические переменные.

**3.1.9 Показатели «открытость»**

* Для выделения текста и его форматирования использована табуляция и автоматическое форматирование, согласно современным методологиям.
* Для избегания неопределенностей приоритетности операций использованы скобки.
* В одной строке кода выполняется одно присваивание.
* В одной строке кода выполняется один оператор.

**3.2 Расчетные показатели качества**

**3.2.1 Показатели «надежность»**

* Устойчивость к искажающим воздействиям.

Из искажающих воздействий использовалось изменение параметров входящих данных, а также имитация их частичной утраты. Было проведено 10 экспериментов, с вводом в различных полях данных. В поле ввода константы обнаружена ошибка. Приводящая к терминальному отказу программы, следовательно:

* Вероятность безотказной работы

Отказов в работе программы при корректном вводе данных не наблюдалось.

* Среднее время восстановления

Qb=1

* Оценка продолжительности преобразований входных данных в выходные

Допустимое время преобразования одиночного полного набора данных = 0.1 секунды.

Среднее время выполнения операции программой = 0.03.

Следовательно Qni = 1.

**3.2.2 Показатели «сопровождаемость»**

* Оценка простоты программы по числу точек входа-выхода
* Оценка простоты по числу условных операторов

**3.3 Экспериментальные показатели качества**

**3.3.1 Показатели «надежность»**

* Возможность обработки ошибочных ситуаций присутствует во всех модулях программы, за исключением модуля ввода константы.
* В программе предусмотрена проверка входных данных на корректность и минимальность для корректной работы программы.
* В программе не предусмотрена проверка полноты выходных данных. Однако при возникновения ошибок при выполнении программы пользователь получит сообщение о сбое.
* Возможность восстановления при сбое информационной системы, процессора – невозможна. При отказе внешних устройств, частично возможна смена вида управления программой.

**3.3.2 Показатели «сопровождаемость»**

* Комментарии присутствуют к каждому модулю в полном объеме, согласно методологии принятой в проекте.
* Отсутствует передача данных из модуля через вызывающий модуль.
* Ограничение на размер модуля отсутствует.
* Количество циклов в коде 3.
* Количество циклов с одним входом и одним выходом – 3.

**3.3.3 Показатели «удобство применения»**

Интерфейс интуитивно понятен и не требует написания дополнительной документации.

В качестве пояснений к визуальной части программы использованы подписи для полей ввода данных.

**3.3.4 Показатели «эффективность»**

* время выполнения ~ 0.03 секунды.
* время реакции на действия пользователя ~ 0.001 секунды.
* Количество функций определяется как «достаточное». Равно 3.
* Программа зависит от операционной системы Windows и установленного пакета .NET Framework.

**3.3.5 Показатели «корректность»**

* Отсутствие ошибок в описании действий пользователя, генерации, настройки.
* Присутствует описание и схема иерархии модулей.

**4. Выводы по работе**

В результате выполнения данной лабораторной работы были изучены методологии оценки качества программного продукта на основе одной из существующих методик.

**5. Список используемых источников**

1. Иванова Г. С. Технология программирования: учебник для вузов, 2-е изд.: – М.: ИД КноРус, 2013. – 333 с.: ил.

2. Объектно-ориентированное проектирование информационных систем. Методические указания к выполнению лабораторных работ – СПб ГУАП, 2020 – 31 c.

**Листинг использованного приложения**

Листинг 1 MainWindow.xaml.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Media;

using System.IO;

using Microsoft.Win32;

namespace Dispatcher

{

/// <summary>

/// Логика взаимодействия для MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

static int pxLen = 60; //длина такта на схеме

public MainWindow()

{

InitializeComponent();

}

public void createTimeLine(int length) //генерация оси тактов

{

int lineLength = (length + 1) \* pxLen;

Border VerticalLine = new Border { Background = Brushes.Wheat, BorderBrush = Brushes.Black, BorderThickness = new Thickness(2), Width = lineLength, Height = 5, Margin = new Thickness(0, 0, 0, 0), HorizontalAlignment = HorizontalAlignment.Left, VerticalAlignment = VerticalAlignment.Center };

this.MyGrid.Children.Add(VerticalLine);

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

{

int marg = i \* pxLen;

Border border = new Border { Background = Brushes.Wheat, BorderBrush = Brushes.Black, BorderThickness = new Thickness(2), Width = 5, Height = 40, Margin = new Thickness(marg, 0, 0, 0), HorizontalAlignment = HorizontalAlignment.Left, VerticalAlignment = VerticalAlignment.Center };

MyGrid.Children.Add(border);

}

}

private void txtTarget\_Drop(object sender, DragEventArgs e) //оработка Drag&Drop

{

string[] files = (string[])e.Data.GetData(DataFormats.FileDrop);

using (StreamReader sr = new StreamReader(files[0]))

{

(sender as TextBox).Text = sr.ReadToEnd();

}

}

private void txtTarget\_DropOver(object sender, DragEventArgs e) //оработка Drag&Drop

{

e.Handled = true;

}

private void Button\_Click(object sender, RoutedEventArgs e)//Обработчик кнопки рассчёт

{

LoadBar.Children.Clear();

ActiveBar.Children.Clear();

solve(Convert.ToInt32(Time.Text));

}

private void Load\_Click(object sender, RoutedEventArgs e)//Обработчик кнопки загрузить

{

OpenFileDialog openFileDialog = new OpenFileDialog();

if (openFileDialog.ShowDialog() == true)

source.Text = File.ReadAllText(openFileDialog.FileName);

}

private void Save\_Click(object sender, RoutedEventArgs e)//Обработчик кнопки сохранить

{

SaveFileDialog saveFileDialog = new SaveFileDialog();

if (saveFileDialog.ShowDialog() == true)

File.WriteAllText(saveFileDialog.FileName, source.Text);

}

void solve(int timeOfCount)

{

List<DispTask> tasks; //лист всех процессов

try

{

tasks = FileManager.getFromString(source.Text); //получаем из поля для ввода

int i = 0; // счетчик тактов

while (tasks.Count != 0) //пока ещё есть невыполненные процессы

{

List<DispTask> loadTaskInTik = new List<DispTask>(); //список задач в готовности

if (i % timeOfCount == 0)

{

loadTaskInTik = tasks.Where(x => x.activateTime <= i).ToList(); //заполняем список

DispTask ActiveTask = loadTaskInTik.Find(y => y.time == loadTaskInTik.Max(x => x.time)); //выбираем активный процесс

loadTaskInTik.Remove(ActiveTask);

TaskBox taskBox = new TaskBox();

taskBox.Content = ActiveTask.name;

taskBox.Width = 55;

taskBox.Margin = new Thickness(0, 0, 5, 0);

ActiveBar.Children.Add(taskBox); //отображаем

ActiveTask.work++; //такт проработал

//выводи список процессов в готовности

StackPanel stackPanel = new StackPanel();

stackPanel.Orientation = Orientation.Vertical;

stackPanel.Margin = new Thickness(0, 0, 5, 0);

if (loadTaskInTik.Count == 0)

{

taskBox = new TaskBox();

taskBox.Content = " ";

taskBox.Width = 55;

stackPanel.Children.Add(taskBox);

}

else

{

foreach (DispTask task in loadTaskInTik)

{

if (task.name != "null")

{

taskBox = new TaskBox();

taskBox.Content = task.name;

taskBox.Width = 55;

stackPanel.Children.Add(taskBox);

}

}

}

LoadBar.Children.Add(stackPanel);

//удаляем процесс, который отраотал своё

if (ActiveTask.work >= ActiveTask.time)

{

tasks.Remove(ActiveTask);

}

}

i++;

}

createTimeLine(i); //рисуем ось

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

}

}

Листинг 2. FileManager.cs

using System;

using System.Collections.Generic;

using System.Windows;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Dispatcher

{

public static class FileManager

{

public static List<DispTask> getFromFile(string path)

{

List<DispTask> tasks = new List<DispTask>();

try

{

using (StreamReader sr = new StreamReader(path))

{

while (!sr.EndOfStream)

{

string str = sr.ReadLine();

var parameters = str.Split(' ');

tasks.Add(new DispTask(parameters[0], Convert.ToInt32(parameters[1]), Convert.ToInt32(parameters[2])));

}

}

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

}

return tasks;

}

public static List<DispTask> getFromString(string str)

{

List<DispTask> tasks = new List<DispTask>();

var strings = str.Split('\n');

foreach (string i in strings)

{

var param = i.Split(' ');

tasks.Add(new DispTask(param[0], Convert.ToInt32(param[1]), Convert.ToInt32(param[2])));

}

return tasks;

}

}

}

**Скриншоты использованного приложения**

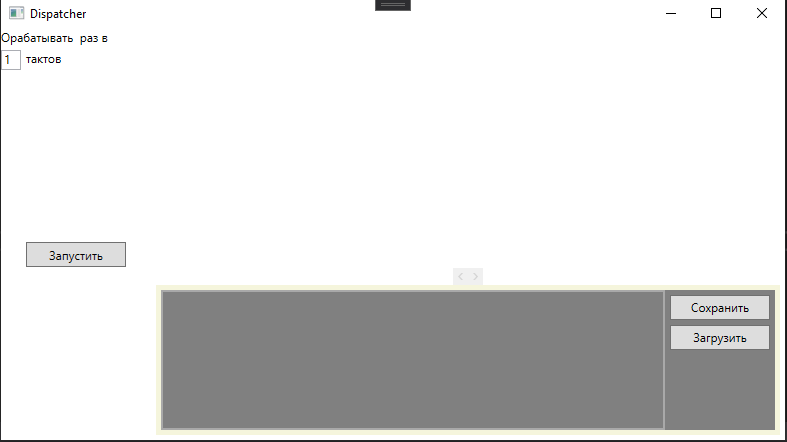


Рисунок 1. Начальное окно

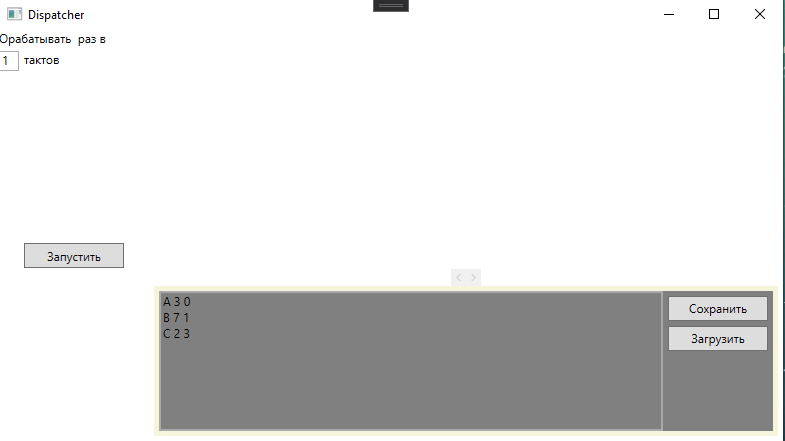


Рисунок 2. Загруженный файл с примером

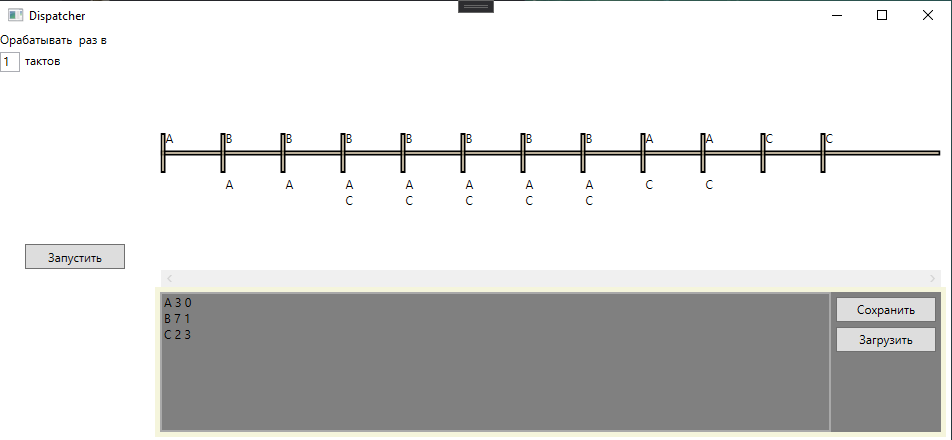


Рисунок 3. Выполненный расчёт