Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Дисциплина: Конструированиепрограмм и языки программирования

К ЗАЩИТЕ ДОПУСТИТЬ

\_\_\_\_\_\_\_\_\_\_\_ Е. В. Калабухов

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему

«Программное средство для подбора конфигурации компьютера»

БГУИР КП 1-40 02 01 617 ПЗ

Студент: Шостак М.Д.

Руководитель: Ассистент кафедры ЭВМ

Юревич А.С.

Минск 2020

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

УТВЕРЖДАЮ

Заведующий кафедрой

*––––––––––––––––––––––––*

(подпись)

–––––––––––––––––––––––– 2020 г.

ЗАДАНИЕ

по курсовому проектированию

Студенту    *Шостаку Михаилу Дмитриевичу–––––––––––*

1. Тема проекта   «Программное средство для подбора конфигурации компьютера»

2. Срок сдачи студентом законченного проекта–––––  *20 декабря 2020 г.–––*

3. Исходные данные к проекту *price.csv (хранятся данные о комплектующих, имеющихся в данный момент в наличии),IDE – Visual Studio 2019, язык программирования – C++*

4. Содержание расчетно-пояснительной записки (перечень вопросов, которые подлежат разработке)

*Введение. 1. Постановка задачи. 2. Структура входных и выходных данных.*

*3. Описание классов. 4. Разработка алгоритмов 4.1.  Схема алгоритмов. 4.2. Алгоритмы по шагам. 5. Код программы 6. Диаграмма классов 7. Результаты работы программы 8. Руководство пользователя. Заключение. Литература.*

5. Перечень графического материала (с точным обозначением обязательных чертежей и графиков)

*1. Схема структурная*

*2. Схема алгоритма* void OutputCostByDetailByChoosenDetail(vector<T>& v, System::Windows::Forms::ComboBox^ cmbbx, System::Windows::Forms::TextBox^ txtbx); *(Вывод цены за выбранную пользователем комплектующую)*

*3. Схема алгоритма* void printNameAndCost(System::Windows::Forms::ComboBox^ comboBox, System::Windows::Forms::TextBox^ txtBox, T& det, int i); *(Вывод названия и цены комплектующей, выбранной программой)*

*4. Диаграмма классов.*

*5. Скриншоты работающей программы––––*

6. Консультант по проекту (с обозначением разделов проекта)  *Юревич А.С.*

7. Дата выдачи задания *14 сентября 2020 г.–––––– –––––––––––––*

*–*

8. Календарный график работы над проектом на весь период проектирования (с обозначением сроков выполнения и трудоемкости отдельных этапов):

*разделы 1,2 к 05.10. – 10 %;––––––––––––––––––––––––––––––––––– –––*

*раздел 3 к 25.10. – 15 %;––––––––––––––––––––––––––––––––––––– –– –*

*раздел 4 к 07.11. – 15 %; –––––––––––––––––––––––––––––––––––––––*

*разделы 5,6 к 19.11. – 25 %;––––––––––––––––––––––––––––––––––––––\_\_\_\_*

*раздел 7 к 26.11. – 25 %;–––––––––––––––––––––––––––––––––––––– –––*

*оформление пояснительной записки и графического материала к 1.12.2020 – 10 %*

*Защита курсового проекта 26.12*

РУКОВОДИТЕЛЬ*–––––––– –––– А. С. Юревич*

(подпись)

Задание принял к исполнению *–––––––\_\_\_\_\_\_ \_\_\_\_\_\_\_ \_– М. Д. Шостак*

(дата и подпись студента)

**СОДЕРЖАНИЕ**

ВВЕДЕНИЕ..............................................................................................................5

1 ОБЗОР ЛИТЕРАТУРЫ…....................................................................................7

2 СИСТЕМНОЕ ПРОЕКТИРОВАНИЕ................................................................9

3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ.................................................11

4 ПРОЕКТИРОВАНИЕ ПРОГРАММНЫХ МОДУЛЕЙ....................................13

5 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ.................................................................15

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ.............................................18

ЗАКЛЮЧЕНИЕ.....................................................................................................19

ПРИЛОЖЕНИЕ А.................................................................................................20

ПРИЛОЖЕНИЕ Б..................................................................................................21

ПРИЛОЖЕНИЕ В..................................................................................................22

ПРИЛОЖЕНИЕ Г..................................................................................................23

ПРИЛОЖЕНИЕ Д..................................................................................................31

**ВВЕДЕНИЕ**

Ускоряющийся темп жизни современного общества обязывает человека воспринимать и обрабатывать огромные объемы информации. Без этого качества современный обыватель не может реализовать себя на все сто процентов. Эта идея сыграла решающую роль при выборе автором темы для курсового проекта. Очень часто встречаются ситуации, когда продавцу, работающему онлайн, пишут сразу несколько клиентов. Тогда перед ним встает дилемма: «Постараться ответить сразу всем или проигнорировать клиентов в пользу кого-то одного?» В первом случае продавец жертвует качеством и рискует потерять сразу всех клиентов, а во втором – получить дополнительную прибыль. Приложение, написанное в ходе курсового проекта призвано решить эту проблему и сделать возможным продавцу комплектующих компьютера поддерживать диалог сразу с несколькими клиентами, ведь оно позволяет ему составлять конфигурации всего за пару кликов!

В основе данного проекта находится объектно-ориентированное программирование, которое представляет собой технологию программирования, базирующуюся на классификации и абстракции объектов. Одним из наиболее популярных средств объектно-ориентированного программирования, позволяющим разрабатывать программы, эффективные по объёму кода и скорости выполнения является С++.

С++ — компилируемый, статически типизируемый язык общего назначения. Он поддерживает такие парадигмы программирования, как процедурное программирование, объектно-ориентированное программирование, обобщённое программирование. Язык имеет богатую стандартную библиотеку, которая включает в себя распространённые контейнеры и алгоритмы, ввод-вывод, регулярные выражения, поддержку многопоточности и другие возможности. С++ сочетает в себе как возможности низкоуровневых языков программирования, так и возможности высокоуровневых. Данный язык программирования был выбран автором из-за его скорости.

Универсальность и гибкость языка позволяют использовать его в различных целях. Область применения данного языка включает создание операционных систем, разнообразных прикладных программ, драйверов устройств, приложений для встраиваемых систем, высокопроизводительных серверов, а также развлекательных приложений (игр). Исходя из этого можно считать, что данный язык достаточно удобен для написания выбранной курсовой работы.

Для работы с графической составляющей игры была выбрана библиотека WinForms. Данная библиотека проста в освоении и предоставляет все необходимые инструменты: создание необходимых элементов интерфейса, регулирование их размеров и положения, обеспечение удобного взаимодействия программы и пользователя, а также вывод на экран.

**1 ОБЗОР ЛИТЕРАТУРЫ**

* 1. **Анализ аналогов программного средства**

Автором были найдены следующие аналоги, находящиеся на

данных интернет-ресурсах:

**https://www.4pc.by/**

**https://www.ironbook.ru/constructor/**

**https://npx.by/conf**

**https://cooler.by/sobrat\_kompjuter\_onlajn.php**

Однако, по мнению автора, эти аналоги уступают в следующих пунктах:

* Слишком много ненужной информации, что может быть полезно человеку, более-менее разбирающемуся в компьютерах и пытающегося составить себе конфигурацию. Однако для профессионала лишние действия означают потерю драгоценного времени.
* Невозможность посмотреть цену отдельной комплектующей.
* Невозможность сохранить или хотя бы скопировать составленную конфигурацию.
* Наличие ситуации, когда пользователь может составить несовместимую конфигурацию (например, выбрать для процессора сокет AMD, а для мат. платы – сокет 1200).
* Невозможность составления конфигурации для заданной цены.

**1.2 Постановка задачи**

В рамках курсового проекта поставлена задача разработать программный продукт, предоставляющий пользователям функциональность быстрого и качественного составления готовой конфигурации компьютера, считывая данные с прайс-листа. В состав продукта, разрабатываемого в ходе работы, входят: прайс-лист, в котором хранятся данные о всех комплектующих, имеющихся в данный момент на складе, а также непосредственно приложение, при помощи которого пользователь получает доступ к функциональности.

Созданное приложение должно иметь понятный интерфейс, высокую скорость работы, а также возможность дополнительного уточнения параметров конфигурации компьютера.

Для реализации программы используется объектно-ориентированный язык программирования C++ и библиотека WinForms, среда разработки Microsoft Visual Studio 2019. Приложение написано для ОС Windows 10.

**2 СИСТЕМНОЕ ПРОЕКТИРОВАНИЕ**

**2.1 Сторонние программные компоненты**

**Windows Forms** — интерфейс программирования приложений (API), отвечающий за графический интерфейс пользователя и являющийся частью Microsoft .NET Framework. Данный интерфейс упрощает доступ к элементам интерфейса Microsoft Windows за счет создания обёртки для существующего Win32 API в управляемом коде. Причём управляемый код — классы, реализующие API для Windows Forms, не зависят от языка разработки. То есть программист одинаково может использовать Windows Forms как при написании ПО на C#, С++, так и на VB.Net, J# и др.

Для упрощения разработки будет использоваться свободная кроссплатформенная библиотека **SFML** (*Simple and Fast Multimedia Library)*. SFML содержит ряд модулей для простого программирования игр и мультимедиа приложений. В настоящее время для данной библиотеки доступны следующие модули:

* System — управление временем и потоками.
* Window — управление окнами и взаимодействием с пользователем.
* Graphics — делает простым отображение графических примитивов и изображений.
* Audio — предоставляет интерфейс для управления звуком.
* Network — для сетевых приложений.

В данной работе будут использоваться модули System (базовый модуль необходимый для работы приложений), Window (создание игрового окна, считывание нажатий клавиатуры и положения курсора) и Graphics (загрузка текстур, создание спрайтов и их отрисовка). Библиотека SFML была выбрана из-за своего удобства при работе с окнами, графикой и простотой.

**2.2 Структура приложения**

В приложении можно выделить несколько основных элементов:

блок считывания исходных данных, блок составления конфигурации, блок записи в файл, блок графической оболочки, блок взаимодействия пользователя с программой.

Блок считывания исходных данных отвечает за считывание из прайс-листа данных о комплектующих, подходящих для компьютеров (например, данные о оборудовании для серверов/ноутбуков не считываются) и не являющихся устаревшими на 2020 год. Данный блок задействуется только при запуске приложения.

Блок составления конфигурации отвечает за анализ полученных данных и выдачу оптимальной рабочей конфигурации компьютера. Данный блок задействуется после нажатия кнопки «Составить конфигурацию».

Блок записи в файл выполняет свою логику после нажатия кнопки «Сохранить конфигурацию в Excel» и записывает полученную конфигурацию в Excel-файл.

Блок графической оболочки отвечает за создание главной формы и отрисовку всех графических элементов(TextBox’ы, выпадающие списки, кнопки, переключатели).

Блок взаимодействия пользователя с программой отвечает за обработку нажатий пользователя, а также за вывод всей необходимой для пользователя информации.

Структурная схема представлена в приложении А.

**3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ**

Диаграмма классов представлена в приложении Б

**3.1 Описание хранения данных программы**

Данные программы хранятся в excel-файле, имеющем определенную структуру.

Первый файл под названием Price.csv хранит данные (ID, название и цену) о комплектующих, находящихся в данный момент в наличиии. Данный файл автоматически открывается программой в самом начале работы программы.

При отклонении от указанной структуры файла программа также оповестит пользователя.

**3.2 Описание работы классов**

**3.2.1 Класс Detail**

Содержит основную информацию, присущую любой комплектующей, необходимой для нормальной работы компьютера, является предком для всех комплектующих. Содержит следующие поля:

* Cost – используется для всех комплектующих, хранит цену данной комплектующих, измеряемую в у.е.
* Name – используется для всех комплектующих, хранит название данной комплектующих, используется программой для дальнейшей обработки информации о данной детали.
* id – показывает, каким по счету была считана данная комплектующая.
* indexInVector – показывает, каким по счету среди всех других деталей этого же типа была считана данная комплектующая.
* Count – используется для инициализации поля id.

Для работы используются следующие методы:

* setCost – установление цены объекта.
* getCost – получение цены объекта.
* setName – установление имени объекта.
* getName – получение имени объекта.
* getId – получение ID объекта.
* getIndex – получение индекса объекта.
* getCount – получение количества всех комплектующих.

**3.2.2 Класс CPU**

Данный класс наследуется от абстрактного класса Detail и отличается от последнего лишь возможностью создания объектов класса.

**3.2.3 Класс MB**

Данный класс наследуется от абстрактного класса Detail и отличается от последнего лишь возможностью создания объектов класса.

**3.2.4 Класс Cooler**

Данный класс наследуется от абстрактного класса Detail и отличается от последнего лишь возможностью создания объектов класса.

**3.2.5 Класс mem**

Данный класс наследуется от абстрактного класса Detail, но имеет и дополнительный функционал в виде методов:

* set\_Quantity\_of\_DDR – установление количества планок памяти в данной конфигурации.
* get\_Quantity\_of\_DDR – получение количества планок памяти в данной конфигурации.

**3.2.6 Класс GPU**

Данный класс наследуется от абстрактного класса Detail и отличается от последнего лишь возможностью создания объектов класса.

**3.2.7 Класс ssd**

Данный класс наследуется от абстрактного класса Detail и отличается от последнего лишь возможностью создания объектов класса.

**3.2.8 Класс hdd**

Данный класс наследуется от абстрактного класса Detail и отличается от последнего лишь возможностью создания объектов класса.

**3.2.9 Класс Case**

Данный класс наследуется от абстрактного класса Detail и отличается от последнего лишь возможностью создания объектов класса.

**3.2.10 Класс PSU**

Данный класс наследуется от абстрактного класса Detail и отличается от последнего лишь возможностью создания объектов класса.

**4 ПРОЕКТИРОВАНИЕ ПРОГРАММНЫХ МОДУЛЕЙ**

* 1. **Схема алгоритмов**

4.1.1 Алгоритм вывода цены, при выборе пользователем нового элемента.

Представлен в Приложении Б.

4.1.2 Алгоритм вывода названия и цены для детали, выбранной программой.

Представлен в приложении В.

* 1. **Алгоритмы по шагам**

4.2.1 Для алгоритма по шагам рассмотрим метод предназначенный для правильной обработки изменения цены, при выборе пользователем нового элемента.

Шаг 1: Начало;

Шаг 2: Получение индекса выбранного элемента;

Шаг 3: Если выбирать нечего, то выход из функции;

Шаг 4: Если вернулся индекс, соответствующий некоторому элементу из списка, то присвоение TextBox’у, соответствующему цене данного ComboBox’а цену за данную деталь;

Шаг 5: Вывод пользователю нового значения цены;

Шаг 6: Конец;

4.2.2 Для алгоритма по шагам рассмотрим метод предназначенный для правильного вывода названия и цены для детали, выбранной программой.

Шаг 1: Начало;

Шаг 2: Если программа еще не составляла конфигурацию, то запомнить значения индекса выбранного элемента;

Шаг 3: Если из прайс-листа не удалось считать комплектующую данного типа, то вывод уведомления в соответствующем ComboBox’е;

Шаг 4: Если данный comboBox не пуст, то вывод названия и цены выбранной комплектующей;

Шаг 6: Конец;

* 1. Описание важных модулей программы

Описание взаимодействия с прайс-листом

Для правильной работы программы, необходимо, чтобы в папке с приложением находился файл “price.csv”, содержащий значения, разделенные запятыми. Программа построчно считывает информацию, создавая экземпляры классов, наследованных от абстрактного класса Detail. Если в считанной строке был значок дюйма (“), то эта ситуация также обрабатывается. Устаревшие комплектующие, комплектующие для серверов и ноутбуков, а также дополнительные аксесуары (манипуляор мышь, клавиатура, монитор и т.д.) программой пропускаются.

**5 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ**

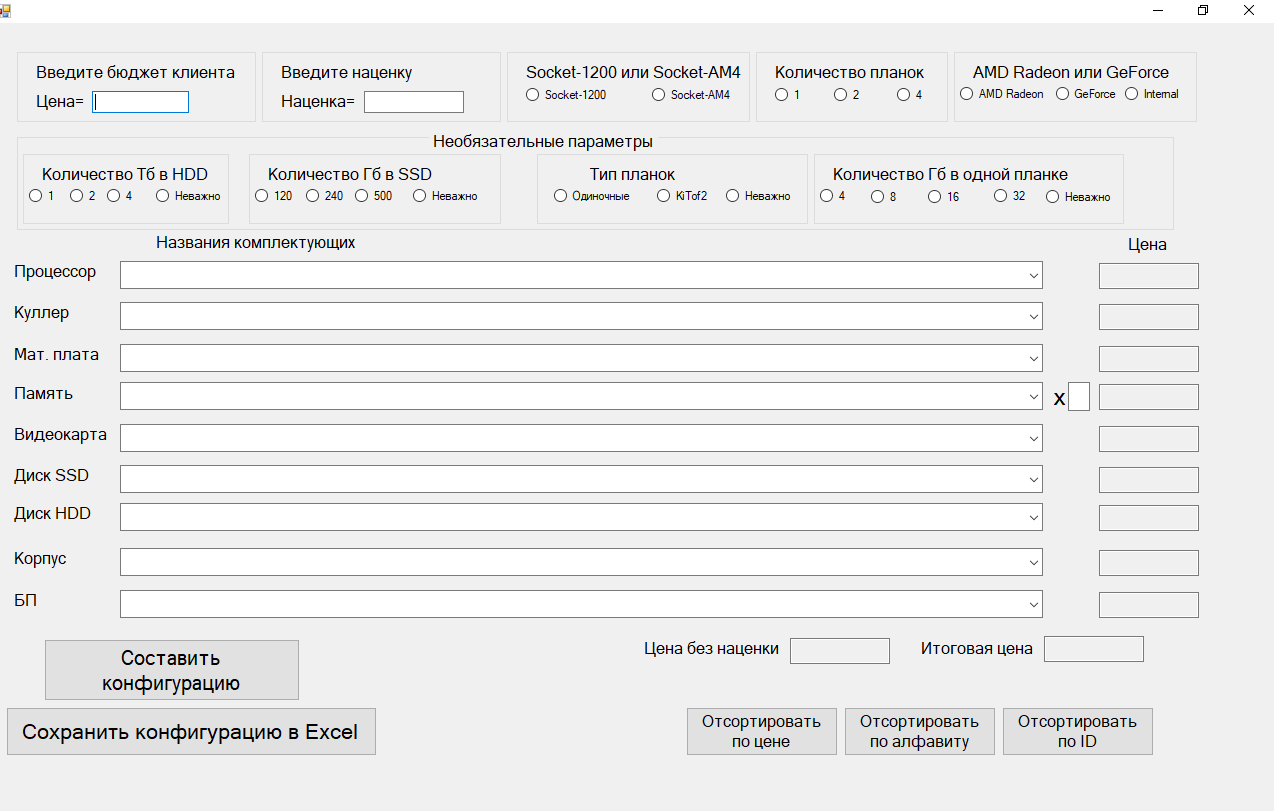
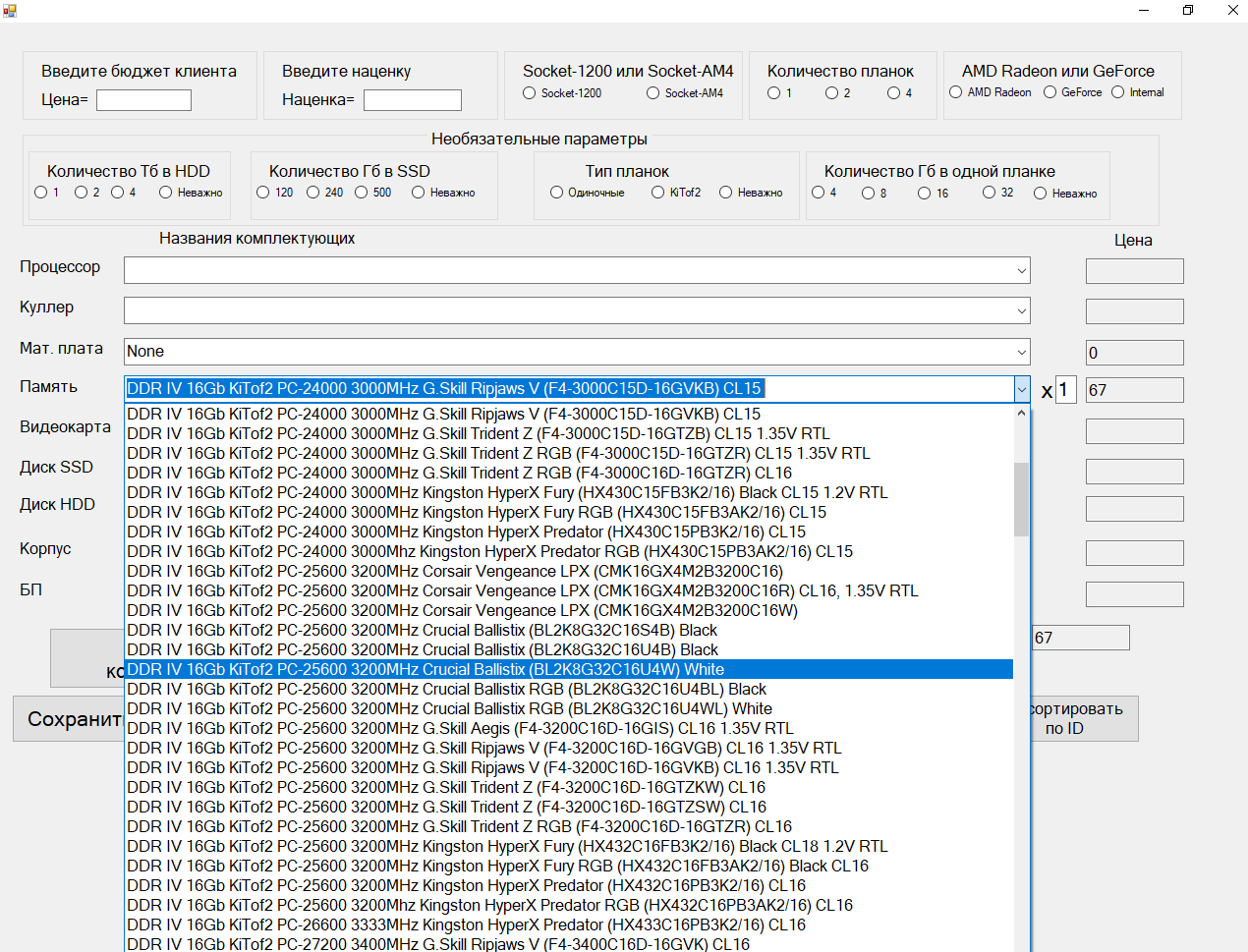
Приложение имеет следующий вид (Рисунок 5.1).

Рисунок 5.1

Пользователю предлагается ввести обязательные параметры, которые необходимы программе для ее успешного выполнения. К ним относятся: бюджет клиента (приблизительная стоимость конфигурации без наценки), наценка (торговая надбавка), выбор типа сокета для процессора, выбор количества планок памяти и выбор типа видеокарты.

Также пользователь может выбрать необязательные параметры, которые служат для более точного составления конфигурации. К необязательным параметрам относятся: выбор количества терабайт в HDD, выбор количества гигабайт в SSD, тип планок памяти (одиночные или в комплекте) и количество гигабайт в одной планке.

Также пользователь может выбрать комплектующую определенного вида из выпадающего списка. Кроме того, реализован выбор пункта «None», которые соответствует отсутствию детали. (Рисунок 5.2)

 Рисунок 5.2

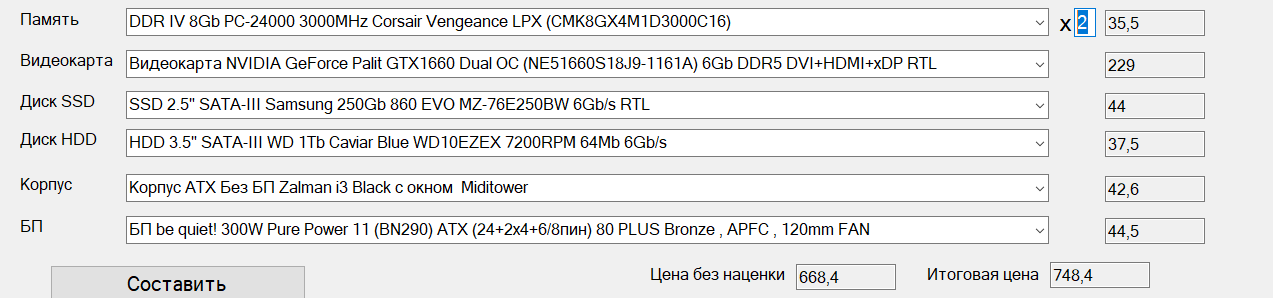
Вдобавок, добавлена возможность изменить количество планок памяти, при этом не составляя заново конфигурацию. Цена (итоговая и с наценкой) автоматически меняются, при изменении комплектующих.

Рисунок 5.3

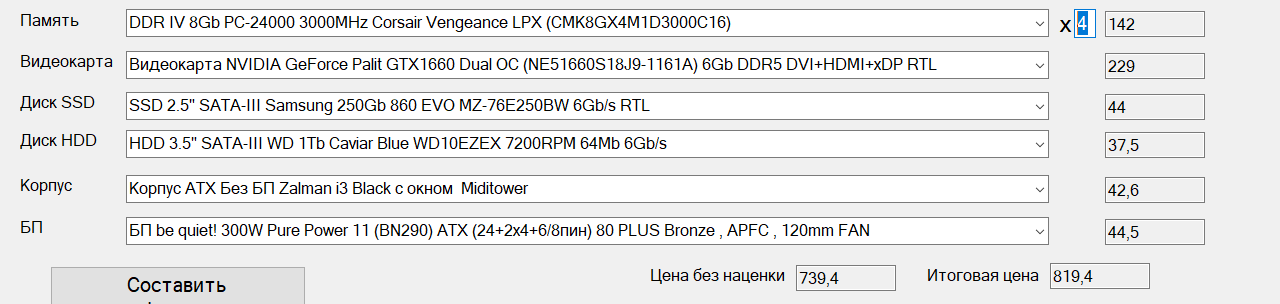


Рисунок 5.4

Данные в списках можно сортировать нажатием соответствующих кнопок:

* «Отсортировать по цене» -- сортирует комплектующие в порядке возрастания цены.
* «Отсортировать по алфавиту» -- сортирует комплектующие в лексикографическом порядке.
* «Отсортировать по ID» -- сортирует комплектующие в том порядке, в котором они были считаны с прайс-листа.

После составления конфигурации, нажав кнопку «Сохранить конфигурацию в Excel» можно сохранить составленную систему в excel-файл. Структура данного файла идентична структуре прайс-листа.

# СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

[1] Страуструп, Б. Язык программирования С++/ Б.Страуструп; специальное издание. Пер. с англ. – Спб.: BHV, 2008. – 1098 с.

[2] METANIT – информационный ресурс для разработчиков [Электронный ресурс]. – Режим доступа: https://metanit.com/

**ЗАКЛЮЧЕНИЕ**

В ходе выполнения данного курсового проекта было разработано десктопное приложение, а также изучено и успешно использовано большое количество текстовой и графической информации, в результате чего стало возможным проектирование программного продукта, его тестирование и устранение ошибок.

Программа позволяет быстро и эффективно составлять конфигурации вручную, или с использованием основной логики программы, а также проверять наличие комплектующих.

В будущем возможно расширение возможностей программы приложения.

**ПРИЛОЖЕНИЕ А**

(обязательное)

Структурная схема.

**ПРИЛОЖЕНИЕ Б**

(обязательное)

Диаграмма классов.

**ПРИЛОЖЕНИЕ В**

(обязательное)

**ПРИЛОЖЕНИЕ Г**

(обязательное)

Код программы

**Detail.h**

#pragma once

#include <string>

#include <vector>

#include <iostream>

#include <fstream>

#include <Windows.h>

using namespace std;

class Detail {

protected:

double Cost;//цена комплектующей

string Name;//название комплектующей

int id;//id комплектующей (каким по порядку считана из документа)

int indexInVector;//индекс комплектующей (как id, только index для каждого вектора свой)

static int indexStatic;//статическая переменная, которую мы будем присваивать полю index элементов, а также обнулять при переходе на новый вектор

static int Count;//общее количество считанных комплектующих

public:

Detail();

Detail(string Name, double Cost);

Detail(const Detail& other);

void setCost(double Cost);

void setName(string Name);

static void setStaticIndexToZero();

double getCost();

virtual void uslessFunction() = 0;//эта функция нужна лишь для того, чтобы сделать класс Detail абстрактный, поэтому, ниже, в её реализации ничего не пишем

string getName();

int getId();

int getIndex();

static int getCount();

bool operator ==(const Detail& other);

bool operator !=(const Detail& other);

bool operator <(const Detail& other);

bool operator >(const Detail& other);

friend ostream& operator<<(ostream& out, const Detail& detail);

friend istream& operator>>(istream& in, Detail& detail);

~Detail();

};

class CPU : public Detail {

public:

virtual void uslessFunction() override {}

CPU(string Name, double Cost) : Detail(Name, Cost) {}

CPU() : Detail() {}

};

**Detail.cpp**

#include "Detail.h"

int Detail::Count = 0;

int Detail::indexStatic = 0;

double Detail::getCost() { //геттеры

return Cost;

};

string Detail::getName() {

return Name;

}

int Detail::getId() {

return id;

}

int Detail::getCount() {

return Count;

}

int Detail::getIndex() {

return indexInVector;

}

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

void Detail::setCost(double Cost) {

this->Cost = Cost;

}

void Detail::setName(string Name) {

this->Name = Name;

}

void Detail::setStaticIndexToZero(){

Detail::indexStatic = 0;

}

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

Detail::Detail() { //конструктор по умолчанию

this->Cost = 0;

this->Name = "empty name";

//this->indexInVector = indexStatic++;

this->id = ++Count;

};

Detail::Detail(string Name, double Cost) { //конструктор с параметрами

this->Name = Name;

this->Cost = Cost;

this->indexInVector = indexStatic++;

this->id = ++Count;

};

Detail::Detail(const Detail& other) { //конструктор копирования

this->Name = other.Name;

this->Cost = other.Cost;

this->indexInVector = other.indexInVector;

this->id = other.id;

};

Detail::~Detail() { //деструктор

this->Name = "empty name";

this->Cost = 0;

this->indexInVector = -1;

this->id = -1;

};

bool Detail:: operator ==(const Detail& other) {

if (this->Cost == other.Cost)

return true;

else return false;

}

bool Detail:: operator !=(const Detail& other) {

if (!(this->Cost == other.Cost))

return true;

else return false;

}

bool Detail:: operator <(const Detail& other) {//не нужны, но пусть будут

if (this->Cost <= other.Cost && this->id <= other.id && this->Name <= other.Name)

return true;

else return false;

}

bool Detail:: operator >(const Detail& other) {

if (this->Cost >= other.Cost && this->id >= other.id && this->Name >= other.Name)

return true;

else return false;

}

ostream& operator<<(ostream& out, const Detail& detail) {

cout << "Name: " << detail.Name << "\nCost: " << detail.Cost << "\nid: " << detail.id << endl;

return out;

};

istream& operator>>(istream& in, Detail& detail) {

in >> detail.Cost;

in >> detail.Name;

return in;

};

void mem::set\_Quantity\_of\_DDR(int quantity) {

quantity\_of\_DDR = quantity;

}

double mem::get\_Quantity\_of\_DDR() {

return quantity\_of\_DDR;

}

int mem::quantity\_of\_DDR = 1;

**main**

template <class T>

string ForConf(double budget, double orientZena, vector<T>& v, T& Det, string comment, string comment2 = " ", string comment3 = " ",

string comment4 = " ", string comment5 = " ") {//передаем бюджет и вектор

//comment2: в случае процессора "AMD" может встречаться и в других комплектующих

//если у нас не процессор, то приравнивание к " " не сломает логику

double raznica, predRaznica = 999999999999;

int frequency\_power, j, k, c = 0, quantity\_of\_proper\_details = 0, quantity\_of\_DDR = mem::get\_Quantity\_of\_DDR();

string name, type;

if (quantity\_of\_DDR == 1 && comment == "Soc-AM4" || comment == "Soc-1200")//минимальное кол-во планок в MB равно 2,

quantity\_of\_DDR = 2; //поэтому на время поиска MB мы минимальное кол-во требуемых планок ставим на 2

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

j = k = 0;

double cost = v[i].getCost();

name = v[i].getName(); //запоминаем название, чтобы не вызывать постоянно метод getName

if (comment == "DDR IV") {//если сейчас рассматриваются планки памяти

if (comment3 == "KiTof2" && name.find("KiTof2") == -1)//если был выбран тип "KiTof2",

continue;//то пропускаем все планки, в чьем имени нет "KiTof2"

else if (comment3 == "Одиночные" && name.find("KiTof2") != -1)

continue;//если тип планок "Одиночные", то пропускаем все планки "KiTof2"

if (comment4 != " " && name.find(comment4+"Gb") == -1)//если было выбрано кол-во Гб в планке

continue;//то если в рассматриваемой планке нет именно такого кол-ва Гб, то мы её пропускаем

}

if (comment2 == "SSD") {//если пользователь ввел нужное ему кол-во Гб в SSD, то пропускаем все, что не подходит

if(comment3 == "120Gb" && name.find("120Gb") == -1)

continue;

if (comment3 == "240Gb" && name.find("240Gb") == -1)

continue;

if (comment3 == "500Gb" && name.find("500Gb") == -1)

continue;

}

if (comment2 == "HDD") {//если пользователь ввел нужное ему кол-во Гб в SSD, то пропускаем все, что не подходит

if (comment3 == "1Tb" && name.find("1Tb") == -1)

continue;

if (comment3 == "2Tb" && name.find("2Tb") == -1)

continue;

if (comment3 == "4Tb" && name.find("4Tb") == -1)

continue;

}

if (budget <= 400 && comment == "ATX" || comment == "MicroATX")//если комп дешевый и мы выбираем корпус

if (name.find("W ") == -1)//выбираем только корпуса со встроенным блоком питания (если в названии есть W (ватты)) (существенная экономия)

continue;//все остальные корпуса пропускаем

if (comment == "Процессор" && comment2 == "Socket-1200" && comment3 == "Internal")//если видеокарта встроенная

if (name.find("F ") != -1)//то пропускаем все процы, которые БЕЗ встроенной видеокарты (для интелов это так)

continue;

if (comment == "Процессор" && comment2 == "Socket-AM4" && comment3 == "Internal")//если видеокарта встроенная

if (name.find("G ") == -1)//ищем, (...G), т.е. со АМДшный проц со встроенной видеокартой

continue;//если ...G нет, то этот проц нам не подходит, ищем следующий

if (comment == "Процессор" && orientZena < 60 && name.find("Процессор BOX") == -1)

continue;//для дешевых конфигураций, чтобы сэкономить, можно купить процессор со встроенным кулером

if (comment == "Socket-AM4" || comment == "DDR IV" || comment == "БП " || comment == "HDD " || (comment == "Universal" && comment2 != " ")) {

if (comment2.find("X ") != -1 && comment == "Universal" && v[i].getCost() < 18)//Если в названии процессора AMD есть X, то кулер нужен помощнее

continue;

if (comment == "DDR IV")//ищем соответствующие параметры для текущей детали

j = name.find("MHz ");

if (comment == "БП ")

j = name.find("W ");

if (comment == "HDD ") {

j = name.find("RPM ");

if (j == -1)//если у нас кол-во оборотов написано маленькими буквами

j = name.find("rpm ");//то ищем маленькие

}

k = j;

if (j == -1)//если у нас в прайс-листе есть какая-то деталь у которой не указаны все необходимые нам параметры

continue;//то мы её просто пропускаем

if (j != 0)

while (isdigit(name[j - 1]))//считываем все число, стоящее перед MHz или W

--j;

frequency\_power = atoi(name.substr(j, k - j + 1).c\_str()); //преобразуем строку в число (находим частоту DDR)

if (budget > 600) {

if ((comment == "DDR IV" && frequency\_power < 3000) //если комп от 600 баксов, то частота DDR минимум 3000

|| (comment == "БП " && (frequency\_power < 600 && name.find("Bronze") == -1))

|| comment == "HDD" && frequency\_power < 7200 && comment != "Universal") //если комп от 600 баксов, то частота DDR минимум 3000 и мощность БП не меньше 600 Вт (и БП должне быть с бронзовым сертификатом), для кулера мы делаем проверку выше

continue;//(просто "дешёвая" конфигурация будет всегда дешевле, чем бюджет, поэтому и 800, а не 600)

} //если не выполняется условие, то переходим к следующей DDR

}

if (comment == "Universal" || (name.find(comment) != -1 && name.find(comment2) != -1)) {//если у нас полное совпадение по желаемым параметрам или кулер типа "Universal" (которому мы всегда рады), то идём дальше

if (comment == "Soc-AM4" || comment == "Soc-1200") { //если мы ищем MB

k = name.find("DDR4");

if (k == -1)

continue;

while (!isdigit(name[k])) //пока не дойдем до кол-ва планок памяти в MB идем к началу строки (там может быть 4хDDR или 4DDR)

--k;

j = k;

while (isdigit(name[j])) //проходим к началу числа, равного максимальному кол-ву DDR в MB (вдруг в прайс-листе будет 16xDDR)

--j;

//++j; //увеличиваем на единицу, так как цикл while дополнительно отнял 1

if (atoi(name.substr(j + 1, k - j).c\_str()) < quantity\_of\_DDR) //если в названии MB максимальное кол-во планок меньше желаемого

continue; //то такие MB мы пропускаем

}

if (comment2 == "1" && (name.find("DDR IV 4Gb PC") != -1 || name.find("KiTof2") != -1)) //одна планка памяти на 4 Гб

continue; //это слишком мало, т.е. нужно чтобы DDR была на 8+ Гб. Также если мы собираем только с 1 планкой памяти, то нельзя KiTof2, т.к. это набор из 2-х планок

if (comment2 == "2" && c == 0 && name.find("KiTof2") != -1) {//если у нас KiTof2, т.е. набор из 2-х планок, то мы увеличиваем цену (которая была для ОДНОЙ планки) в 2 раза

orientZena \*= 2;

c = 1;

}

else if (c == 1) {

orientZena /= 2; //если у нас не набор, то мы возвращаем цену в изначальное состояние

c = 0;

}

if (comment2 == "4" && c == 0 && name.find("KiTof2") != -1) {//если у нас KiTof2, т.е. набор из 2-х планок, то мы увеличиваем цену (которая была для ОДНОЙ планки) в 2 раза

orientZena \*= 2;

c = 1;

}

else if (c == 1) {

orientZena /= 2;

c = 0;

}

++quantity\_of\_proper\_details; //переменная нужна для того, чтобы узнать, у нас нет заданных деталей или бюджета не хватает

raznica = orientZena - v[i].getCost();

if (cost <= orientZena && raznica < predRaznica) {

//для кулера типа "Universal" данный тип может и не быть указанным в названии, но когда мы считывали кулеры,

//то мы считывали только кулеры этого типа,

//т.е. если кулер прошел все проверки, то мы его спокойно пропускаем

//если тип детали совпадает с желаемым и цена текущего экземпляра меньше и максимально близко к ориентеровочной, то запоминаем его

Det = v[i];//запоминаем этот экземпляр в отдельную переменную

if (raznica == 0) //если мы нашли полное совпадение по желаемой цене

return comment;//то дальше нет смысла искать

predRaznica = raznica; //запоминаем прошлую разницу, чтобы допускать только те детали, цена которых более близкая к требуемой

}

}

}

return comment;

}

**ПРИЛОЖЕНИЕ Д**

(обязательное)