Министерство науки и высшего образования Российской Федерации

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ (НИ ТГУ)

Институт прикладной информатики и компьютерных наук

Кафедра прикладной информатики

ОТЧЁТ

по лабораторной работе № 1

по предмету Интеллектуальные системы  
  
Тема: ЭС, рекомендующая конфигурацию персонального компьютера

Выполнили  
студенты 2 курса   
группы № 931902

Макаров С. С. Донской В. Д. Косарев Д. Н.

Содержание

[1. Определение проблемной области 4](#_Toc67709361)

[1.1 Идентификация проблемной области 4](#_Toc67709362)

[Проблемная область 4](#_Toc67709363)

[Подцели 4](#_Toc67709364)

[Требования к разрабатываемой ЭС 4](#_Toc67709365)

[Факторы 4](#_Toc67709366)

[Сфера применения экспертной системы 4](#_Toc67709367)

[Источник знаний 4](#_Toc67709368)

[Примеры проблем 4](#_Toc67709369)

[Критерии оценки 5](#_Toc67709370)

[Выделение ресурсов 5](#_Toc67709371)

[1.2 Цель работы 6](#_Toc67709372)

[Цель 6](#_Toc67709373)

[Желаемые результаты 6](#_Toc67709374)

[Примеры целевой аудитории 6](#_Toc67709375)

[2. Концептуализация проблемной области 7](#_Toc67709376)

[2.1 Построение объектной модели 7](#_Toc67709377)

[2.2 Построение функциональной модели 8](#_Toc67709378)

[Дерево целей 8](#_Toc67709379)

[Дерево решений 8](#_Toc67709380)

[3. Формализация базы знаний 9](#_Toc67709381)

[3.1 Построение логической модели 9](#_Toc67709382)

[Текстовая версия логической модели. 9](#_Toc67709383)

[Графическая версия логической модели 10](#_Toc67709384)

[3.2 Построение продукционной модели на основе нечеткой логики. 10](#_Toc67709385)

[4. Этап реализации ЭС 12](#_Toc67709386)

[Описание метода решения задачи 12](#_Toc67709387)

[Блок-схема алгоритма 13](#_Toc67709388)

[Скриншоты работы программы 14](#_Toc67709389)

[Листинг программы 15](#_Toc67709390)

[5. Этап тестирования 15](#_Toc67709391)

[Оценка точности работы 15](#_Toc67709392)

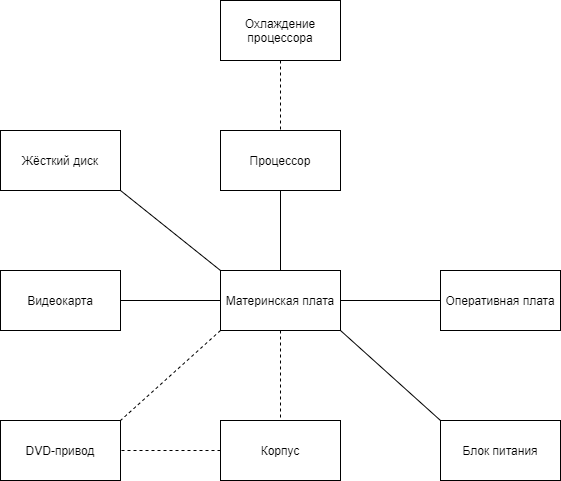
[Вывод 19](#_Toc67709393)

# 1. Определение проблемной области

## 1.1 Идентификация проблемной области

### Проблемная область

Подбор комплектующих для персонального компьютера.



### Подцели

Подбор комплектующих и предоставление готовых решений, соответствующих заданным требованиям по допустимой цене.

### Требования к разрабатываемой ЭС

Автоматизация и оптимизация решения проблемы подбора персонального компьютера, распространение и использования уникального опыта пользователей.

### Факторы

Сфера деятельности, бюджет, предпочтения пользователя.

### Сфера применения экспертной системы

Бытовая, потребительская сфера, ситуация покупки ПК или его комплектующих.

### Источник знаний

В качестве источника знаний мы будем использовать наши собственные знания и информацию из интернет-ресурсов.

(Например: википедия, интернет-магазины, сайты производителей комплектующих)Мы использовали:

<https://www.dns-shop.ru/>

<https://www.youtube.com/>

<https://www.e-katalog.ru/>

<https://www.nvidia.com/>

<https://www.amd.com/>

<https://www.intel.com/>

### Примеры проблем

Пользователь хочет приобрести комплектующие, но не знает, что существуют лучшие альтернативы; Пользователь не знает какая конфигурация ПК будет предпочтительнее для выполнения его задач; Пользователь не знает о всех новинках рынка.

### Критерии оценки

Бэнчмарки (очки рейтинга производительности системы) в специализированных приложениях; параметры, указанные пользователем.

### Выделение ресурсов

Мы выделяем троих экспертов данной области на решение этой задачи. Они уделят время из учебного плана на программирование экспертной системы и подбор вариантов сборок комплектующих.

## 1.2 Цель работы

### Цель

Упростить подбор комплектующих для ПК, сделать максимально доступную программу для любого уровня пользователей, предоставить готовые решения по заданным критериям.

### Желаемые результаты

Нахождение оптимальной конфигурации ПК, подходящей конкретному пользователю. Предоставление альтернатив и возможности выбора пользователю в случае неочевидности оптимальной конфигурации ПК.

### Примеры целевой аудитории

1) Начинающий предприниматель, который завёл малый бизнес, и ему требуется аппаратное обеспечение для выполнения задач внутри предприятия. (например, несколько компьютеров для офиса)

2) Рядовой пользователь, который использует компьютер для серфинга в интернете и для социальных сетей.

3) Высококвалифицированный специалист, компьютер ему необходим для сложных вычислительных операций, моделирования, видеомонтажа или создания сложных проектов.

4) Пользователь, ищущий компьютер для игр. Игры довольно требовательны к производительности системы, тем не менее его запросы ниже, чем у специалиста.

# 2. Концептуализация проблемной области

## 2.1 Построение объектной модели

Объектная модель в нашем случае выражена ER-схемой предметной области.

### Текстовая версия ER-схемы.

**Материнская плата** (Сокет, кол-во слотов под оперативную память, тип памяти, цена),

**Оперативная память** (Тип памяти, частота, объём, цена),

**Блок питания** (Мощность, модульность, цена),

**Корпус** (Размер, охлаждение, цена),

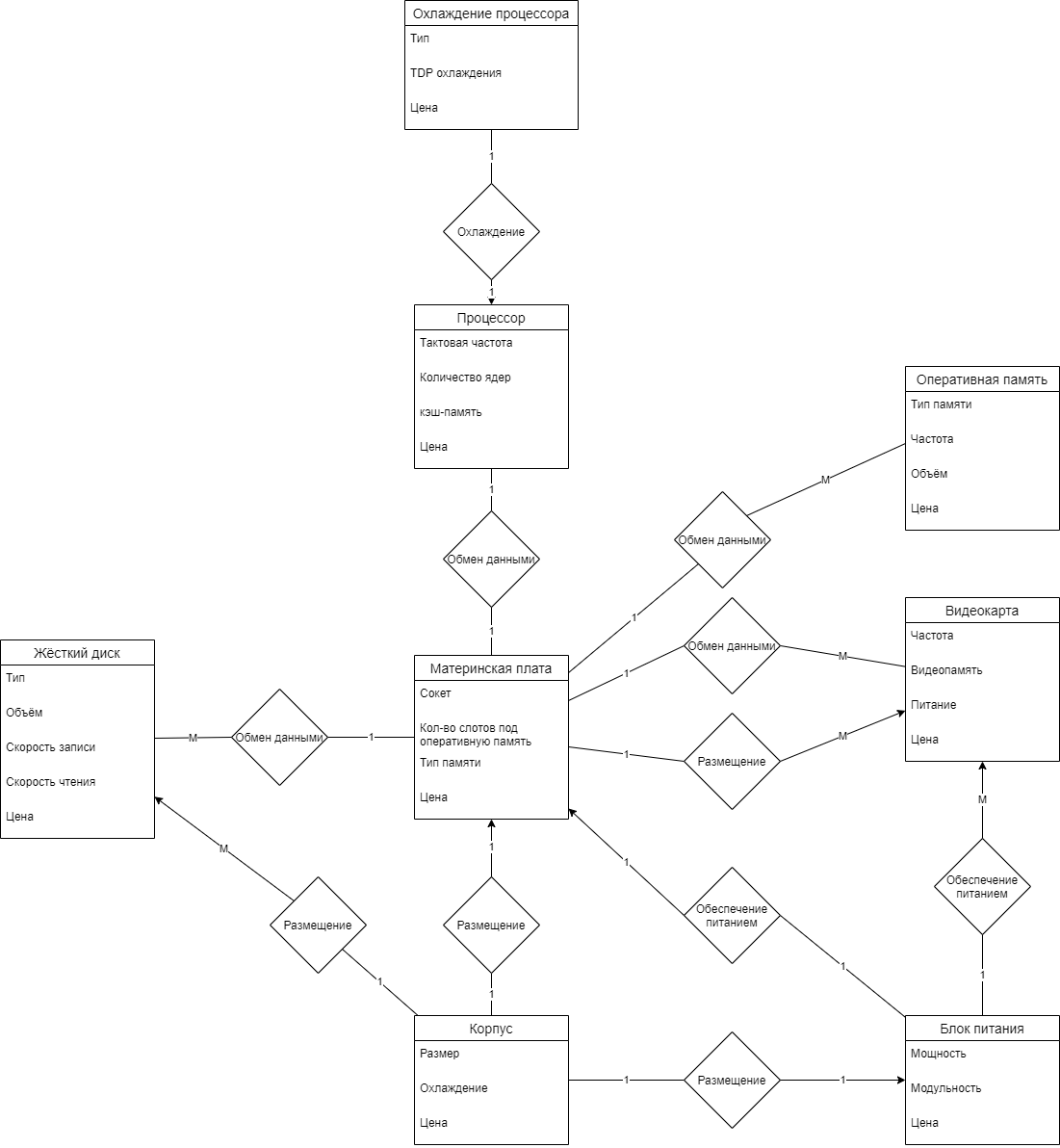
**Видеокарта** (Частота, видеопамять, питание, цена),

**Жёсктий диск** (Тип, объём, скорость записи, скорость чтения, цена),

**Процессор** (Тактовая частота, количество ядер, количество потоков, кэш-память, цена),

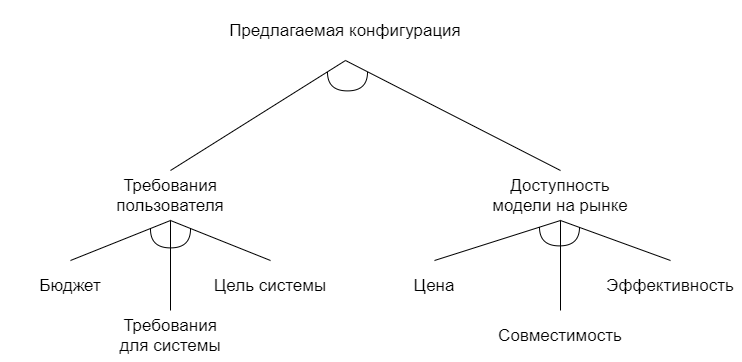
**Охлаждение** (Тип, TDP охлаждения, цена)

### Графическая версия ER-схемы.

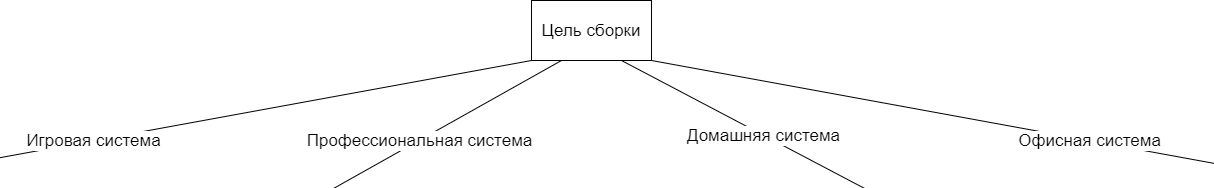


## 2.2 Построение функциональной модели

### Дерево целей



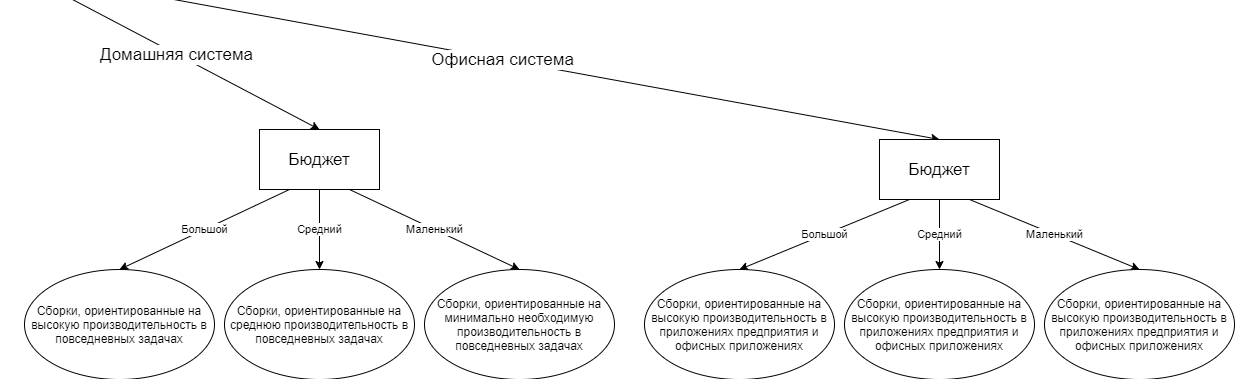
### Дерево решений



Левая половина



Правая половина



# 3. Формализация базы знаний

## 3.1 Построение логической модели

Для оценки эффективности комплектующих мы используем условные баллы.

Бюджет: Низкий, Средний, Большой;

Потребности системы: Домашняя, Офисная, Игровая, Профессиональная

Сборки:

Низкий Домашняя, Средний Домашняя, Низкий Офисная 0 – 40 баллов

Большой Домашняя, Средний Офисная, Низкий Игровой, 31 – 70 баллов

Большой Офисная, Средний Игровая, Низкий Проф 61 – 100 баллов

Большой Игровая, Средний Проф, Большой Проф 91 – 120 баллов

Если требования Домашняя и Бюджет Низкий То Мощность 10

Если требования Домашняя и Бюджет Средний То Мощность 20

Если требования Домашняя и Бюджет Большой То Мощность 40

Если требования Офисная и Бюджет Низкий То Мощность 30

Если требования Офисная и Бюджет Средний То Мощность 50

Если требования Офисная и Бюджет Большой То Мощность 70

Если требования Игровая и Бюджет Низкий То Мощность 60

Если требования Игровая и Бюджет Средний То Мощность 80

Если требования Игровая и Бюджет Большой То Мощность 100

Если требования Проф и Бюджет Низкий То Мощность 90

Если требования Проф и Бюджет Средний То Мощность 110

Если требования Проф и Бюджет Большой То Мощность 120

## 3.2 Построение продукционной модели на основе нечеткой логики.

Правила продукционной модели, работающей с пользователем:

ЕСЛИ указанному пользователем в программе диапазону цены не будет соответствовать ни одна сборка, ТО пользователю будет предложено повысить бюджет или изменить критерии поиска на более дешёвые в реализации.

ЕСЛИ в сборке, которую собирает пользователь самостоятельно, потребление энергии комплектующих превышает количество энергии, выделяемое блоком питания, ТО пользователю будет сказано о невозможности создать данную сборку по причине слабого блока питания.

ЕСЛИ в сборке, которую собирает пользователь самостоятельно, сокет процессора не соответствует сокету материнской платы, ТО пользователю будет сказано о невозможности создать данную сборку по причине несоответствия сокетов.

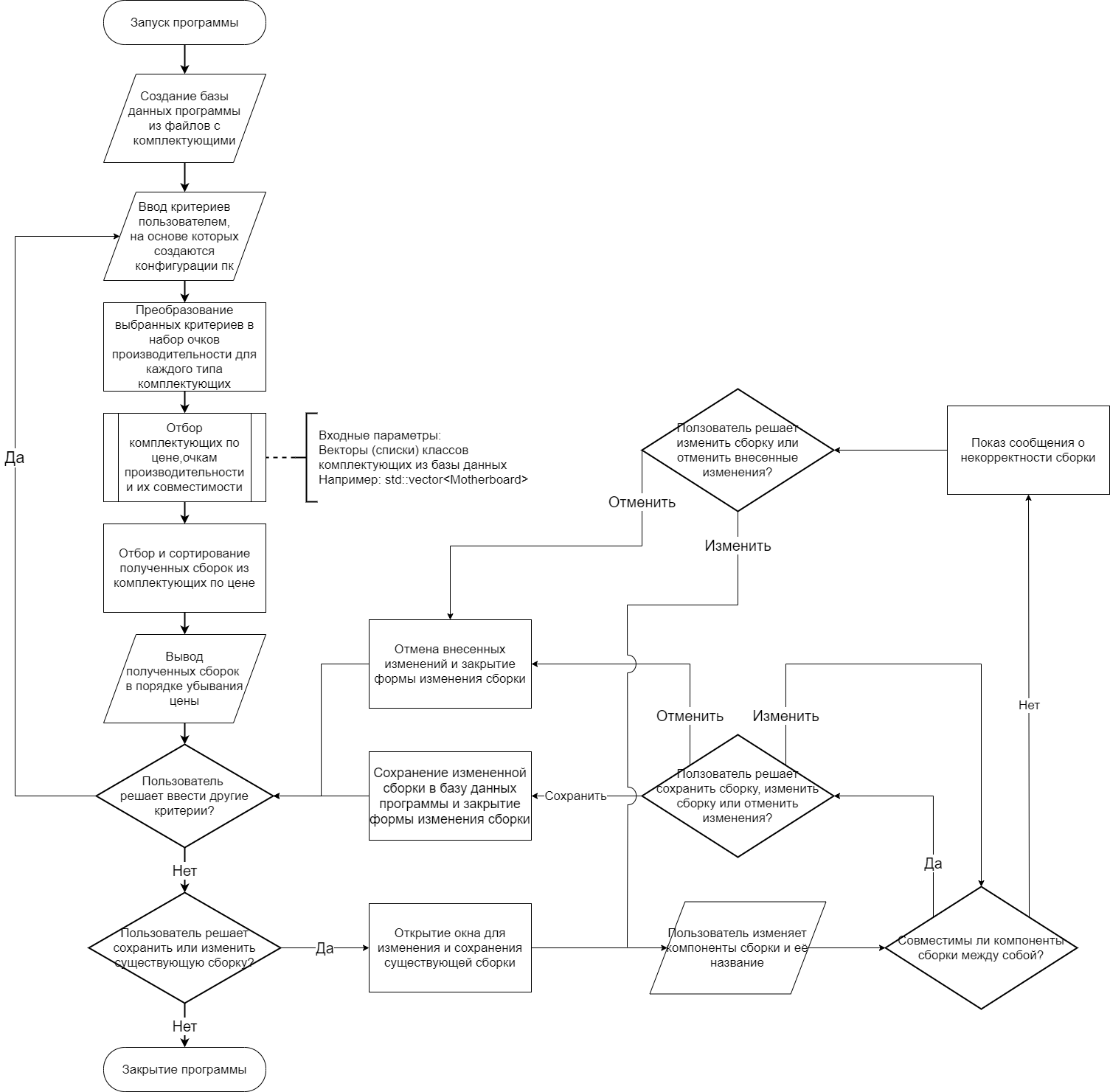
ЕСЛИ в сборке, которую собирает пользователь самостоятельно, производительность процессора и видеокарты значительно различаются (это означает, что либо видеокарта, либо процессор, ограничивают друг друга и урезают общую производительность системы) ТО пользователю будет показано предупреждение, говорящее большой разнице в производительности процессора и видеокарты и сказано какой компонент в данной комбинации намного мощнее.

# 4. Этап реализации ЭС

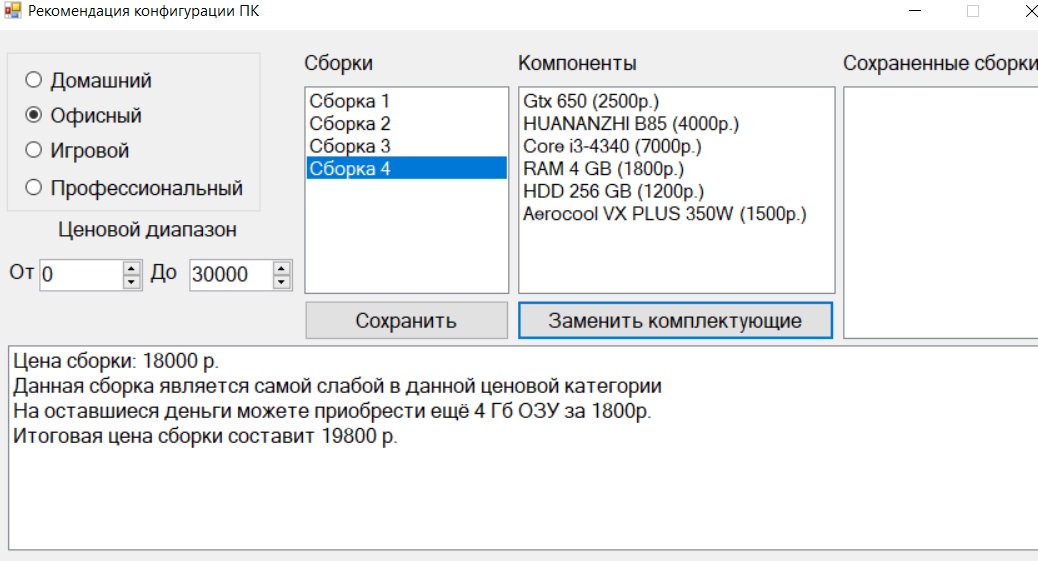
## Описание метода решения задачи

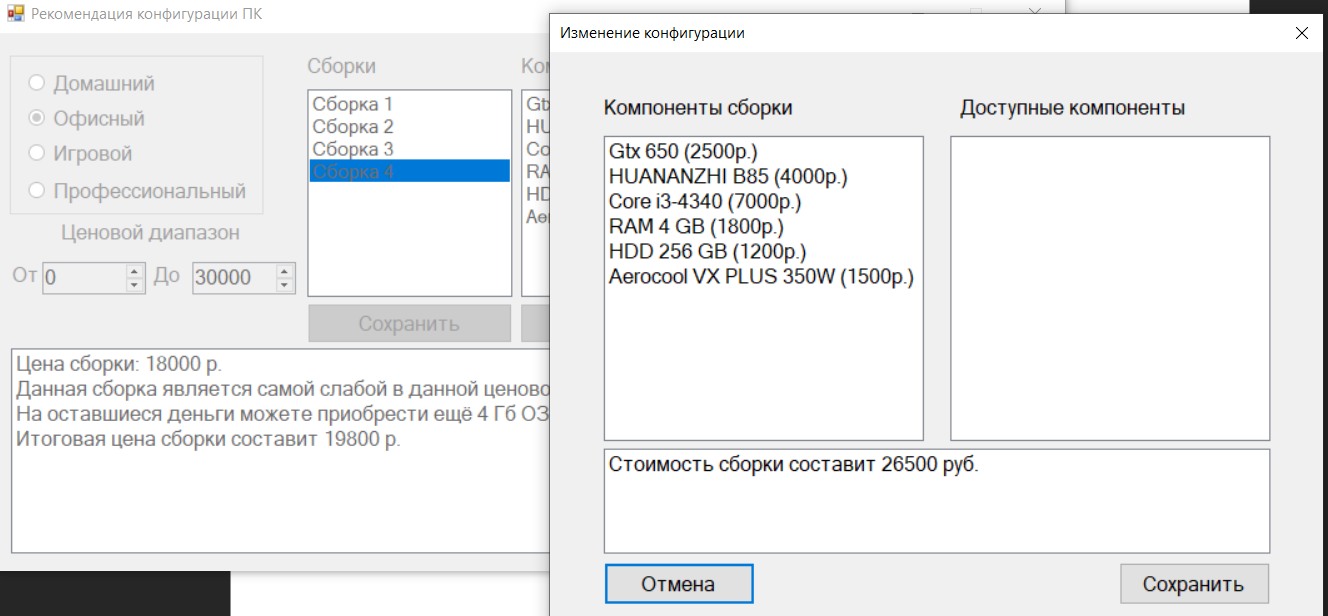
База данных состоит из комплектующих, их характеристик и условных очков производительности. Внутри программы генерируются ограничения сборок по очкам производительности, которые берутся из преобразования критериев пользователя. Затем список сборок, отсортированный по убыванию мощностей, ограниченный бюджетом пользователя выводится на экран. Пользователь может выбрать сборку из предложенных или кастомизировать и сохранить свою сборку.

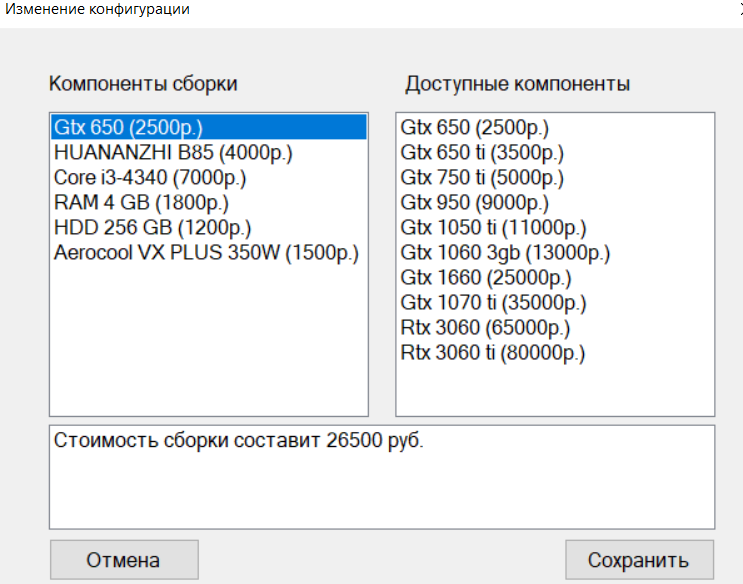
## Блок-схема алгоритма



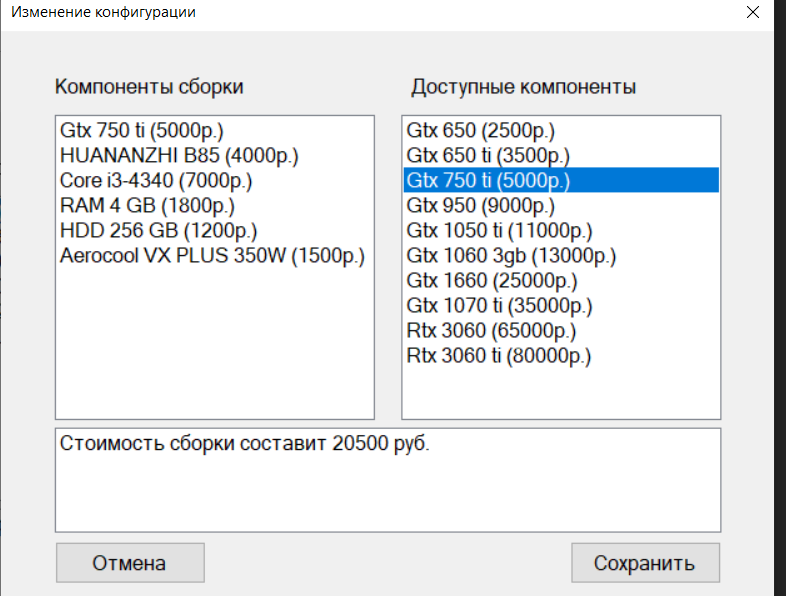
## Скриншоты работы программы

  
Выбор офисной сборки.

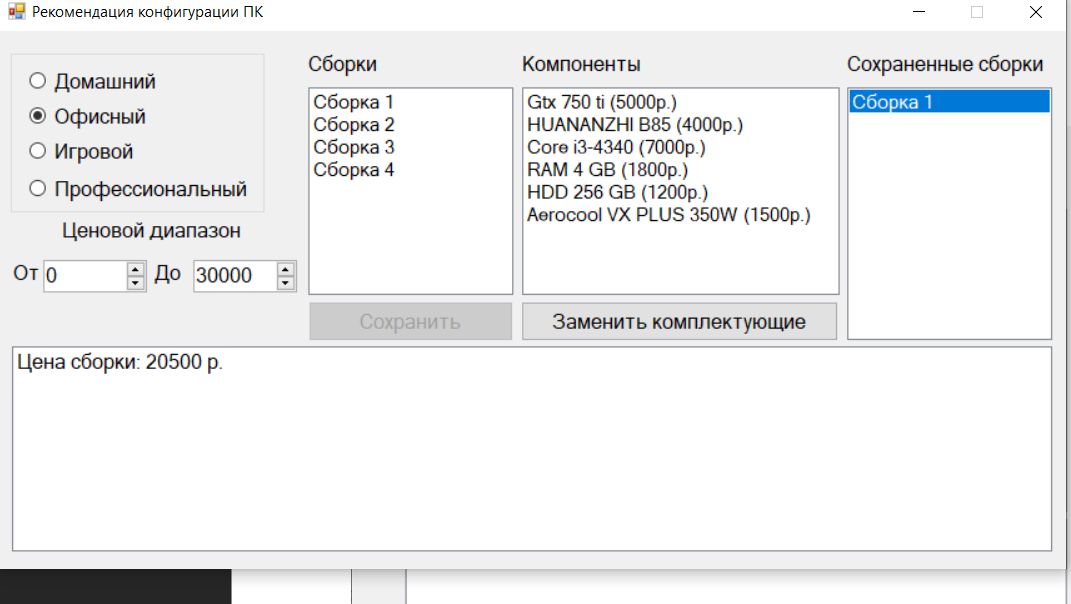


Изменение конфигурации сборки

Просмотр списка доступных видеокарт



Выбор видеокарты

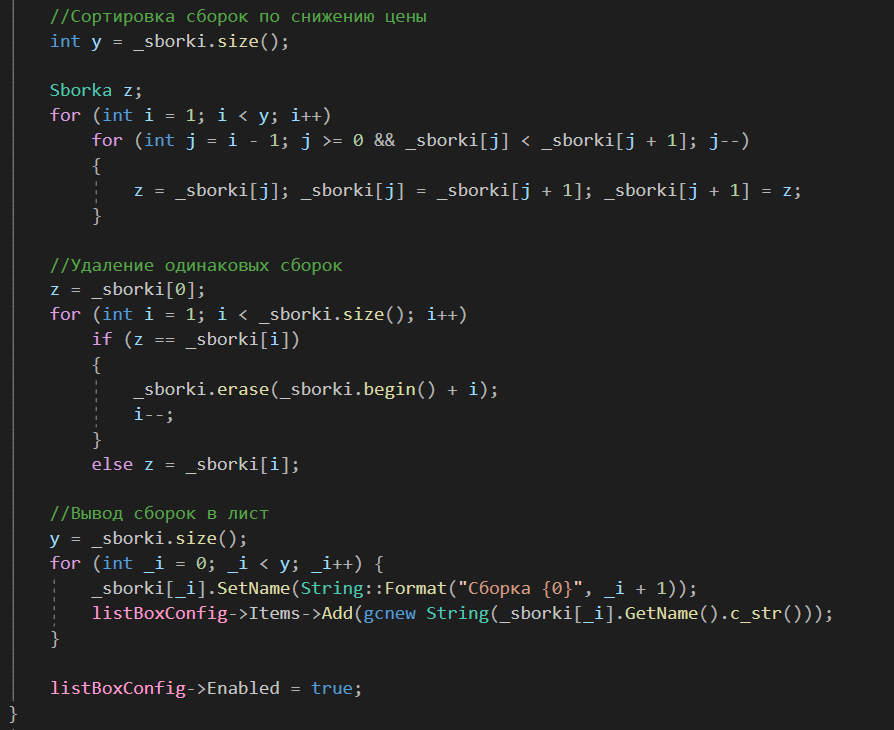
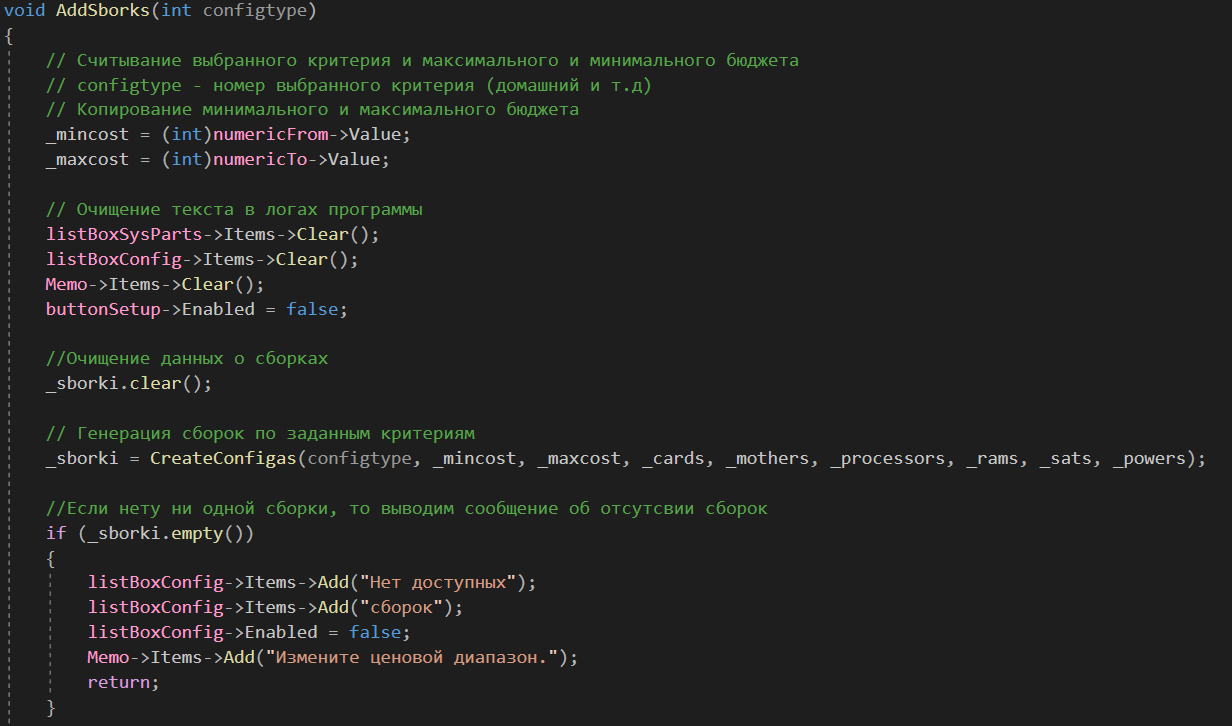


Сборка сохранена

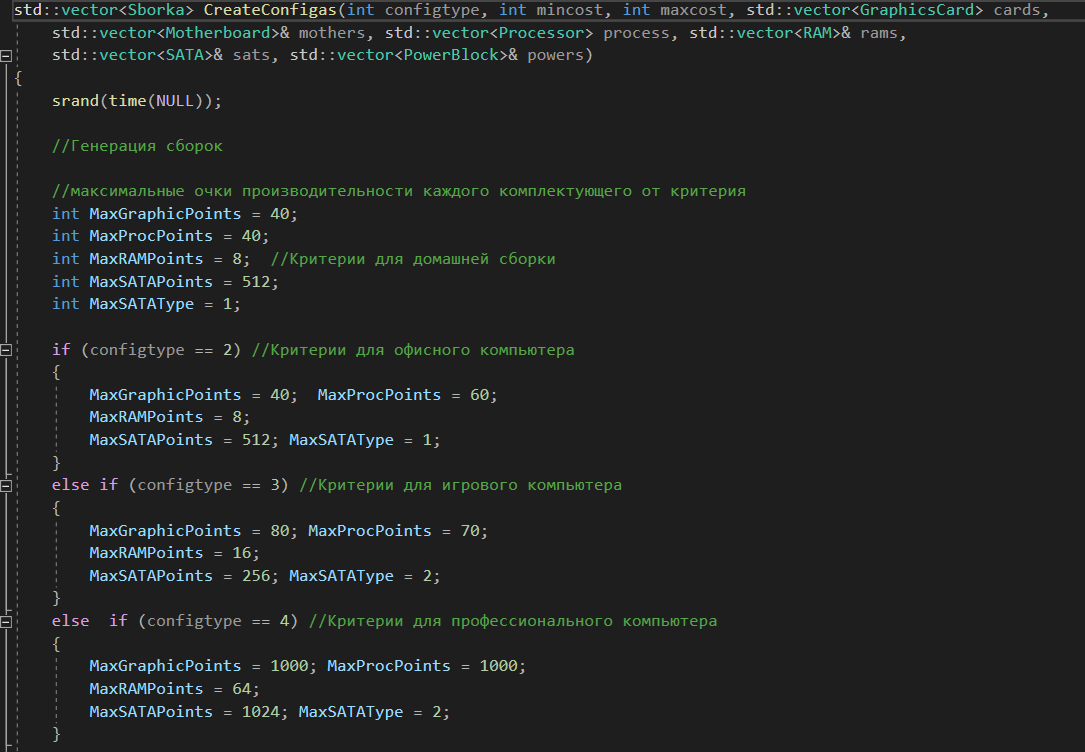
## Листинг программы

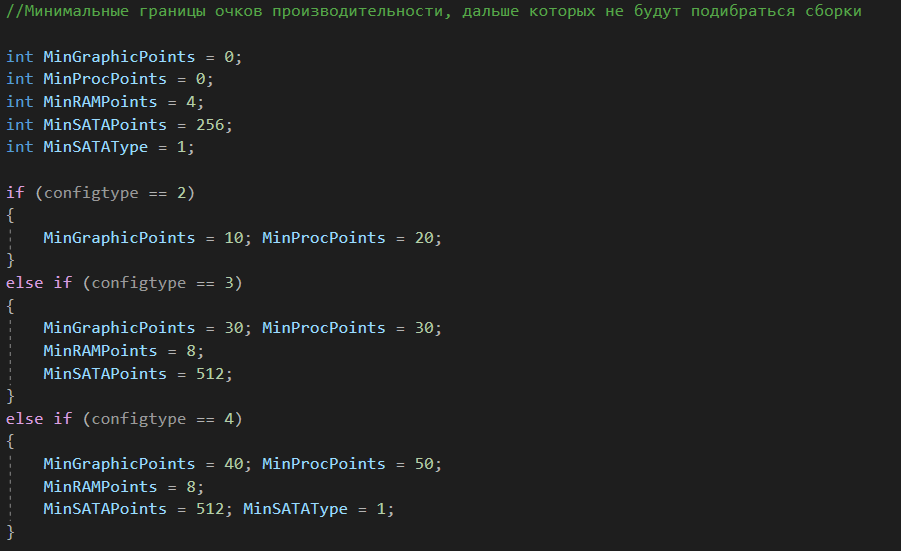
Загрузка формы и данных о комплектующих из файлов

## 

Считывание вводимых пользователем критериев и генерация сборок по ним

Начало генерирования сборок по критериям





//Процессоры, подходящие к видеокарте

std::vector<Processor> ProcRange;

//Поиск видеокарт, подходящих по критериям

for (int i = cards.size() - 1; i >= 0; i--)

if (cards[i].GetPoints() > MaxGraphicPoints) cards.erase(cards.begin() + i);

else if (cards[i].GetPoints() < MinGraphicPoints) cards.erase(cards.begin() + i);

//Поиск процессоров, подходящих по критериям

for (int i = process.size() - 1; i >= 0; i--)

if (process[i].GetPoints() > MaxProcPoints) process.erase(process.begin() + i);

else if (process[i].GetPoints() < MinProcPoints) process.erase(process.begin() + i);

//Для временного хранения выборов комплектации

GraphicsCard card;

Processor proces;

Motherboard mother;

RAM ram;

SATA sata;

PowerBlock power;

//Собранные сборки

std::vector<Sborka> Sborki;

//Текущие данные для циклов сборки

int CurrentRAMPoints = MaxRAMPoints;

int CurrentSATAPoints = MaxSATAPoints;

int CurrentSATAType = MaxSATAType;

int tdp = 0;

//Первая по мощности видеокарта

int GraphicNumber = cards.size() - 1;

card = cards[GraphicNumber];

// Поиск процессоров, подходящих к текущей видеокарте

if (ProcRange.empty())

for (int j = 0; j < process.size(); j++) if (abs(process[j].GetPoints() - card.GetPoints()) <= 20) ProcRange.push\_back(process[j]);

//Генерация сборок

while (true)

{

tdp = 0;

//Запись видеокарты и её энергопотребления

card = cards[GraphicNumber];

tdp += card.GetTdp();

//Запись процессора и его энергопотребления

proces = ProcRange[ProcRange.size() - 1];

tdp += proces.GetTdp() + 175;

//Поиск материнской карты по сокету процессора

int b = 0;

int e = mothers.size();

for (; mothers[b].GetSocket() != proces.GetSocket(); b++);

mother = mothers[b];

//Поиск оперативной памяти

b = 0;

e = rams.size();

while (b < e)

{

int c = (b + e) / 2;

if (rams[c].GetGB() < (int)CurrentRAMPoints) b = c + 1;

else e = c;

}

if (b > (rams.size() - 1)) b = rams.size() - 1;

ram = rams[b];

//Поиск жесткого диска по мощности

b = 0;

e = sats.size();

for (int j = 0; j < e; j++)

if (sats[j].GetGB() == CurrentSATAPoints && sats[j].GetType() == CurrentSATAType)

{

b = j;

break;

}

sata = sats[b];

//Поиск блока питания по энергопотреблению

b = 0;

e = powers.size();

while (b < e)

{

int c = (b + e) / 2;

if (powers[c].GetWatt() < (int)tdp) b = c + 1;

else e = c;

}

power = powers[b];

Sborka sb;

sb.SetConfig(card, mother, proces, ram, sata, power);

if (sb.GetCost() >= mincost && sb.GetCost() <= maxcost)

{

//Сохраняем сборку, если она подходит по цене и собираем следующую сборку

Sborki.push\_back(sb);

//Переходим к следующему процессору, который подходит к выбранной видеокарте, если таковых нет, то меняем видеокарту

if (ProcRange.size() > 1) ProcRange.erase(ProcRange.begin() + ProcRange.size() - 1); //Следующий процессор

else

{

//Следующая видеокарта

cards.erase(cards.begin() + cards.size() - 1);

GraphicNumber = cards.size() - 1;

//Если кончились видеокарты, подходящие по критериям

if (cards.empty()) return Sborki;

else

{

card = cards[GraphicNumber];

ProcRange.clear();

for (int j = 0; j < process.size(); j++) if (abs(process[j].GetPoints() - card.GetPoints()) <= 20) ProcRange.push\_back(process[j]);

}

}

//Сброс критериев вторичных компонентов

CurrentSATAType = MaxSATAType;

CurrentSATAPoints = MaxSATAPoints;

CurrentRAMPoints = MaxRAMPoints;

}

else //Не подошло по цене, уменьшаем не сильно значимые комплекутющие

{

if (CurrentSATAType != MinSATAType || CurrentSATAPoints != MinSATAPoints) //Жесткий диск

{

if (CurrentSATAType == MinSATAType && CurrentSATAPoints > MinSATAPoints)

{

CurrentSATAPoints /= 2;

continue;

}

else if (CurrentSATAType > MinSATAType && CurrentSATAPoints == MinSATAPoints)

{

CurrentSATAType--;

CurrentSATAPoints = 1024;

continue;

}

else if (CurrentSATAType > MinSATAType && CurrentSATAPoints > MinSATAPoints)

{

CurrentSATAPoints /= 2;

continue;

}

else if (CurrentSATAType > MinSATAType && CurrentSATAPoints == 256 && CurrentSATAPoints != MinSATAPoints)

{

CurrentSATAType--;

CurrentSATAPoints = 1024;

continue;

}

}

else if (CurrentRAMPoints > MinRAMPoints)//Опреативная память

{

CurrentRAMPoints /= 2;

continue;

}

else

{

if (ProcRange.size() > 1) //Переходим к следующему процессору, который подходит к выбранной видеокарте, если таковых нет, то меняем видеокарту

{

ProcRange.erase(ProcRange.begin() + ProcRange.size() - 1);

//Сброс критериев вторичных компонентов

CurrentSATAType = MaxSATAType;

CurrentSATAPoints = MaxSATAPoints;

CurrentRAMPoints = MaxRAMPoints;

continue;

}

else //Уменьшаем мощность видеокарты

{

cards.erase(cards.begin() + cards.size() - 1);

GraphicNumber = cards.size() - 1;

//Если кончились видеокарты, подходящие по критериям

if (cards.empty()) return Sborki;

else

{

card = cards[GraphicNumber];

ProcRange.clear();

for (int j = 0; j < process.size(); j++) if (abs(process[j].GetPoints() - card.GetPoints()) <= 20) ProcRange.push\_back(process[j]);

}

//Сброс критериев вторичных компонентов

CurrentSATAType = MaxSATAType;

CurrentSATAPoints = MaxSATAPoints;

CurrentRAMPoints = MaxRAMPoints;

continue;

}

}

}

}

Полная функция создания сборки

void AddSborks(int configtype)

{

// Считывание выбранного критерия и максимального и минимального бюджета

// configtype - номер выбранного критерия (домашний и т.д)

// Копирование минимального и максимального бюджета

\_mincost = (int)numericFrom->Value;

\_maxcost = (int)numericTo->Value;

// Очищение текста в логах программы

listBoxSysParts->Items->Clear();

listBoxConfig->Items->Clear();

Memo->Items->Clear();

buttonSetup->Enabled = false;

//Очищение данных о сборках

\_sborki.clear();

// Генерация сборок по заданным критериям

\_sborki = CreateConfigs(configtype, \_mincost, \_maxcost, \_cards, \_mothers, \_processors, \_rams, \_sats, \_powers);

//Если нету ни одной сборки, то выводим сообщение об отсутсвии сборок

if (\_sborki.empty())

{

listBoxConfig->Items->Add("Нет доступных");

listBoxConfig->Items->Add("сборок");

listBoxConfig->Enabled = false;

Memo->Items->Add("Измените ценовой диапазон.");

return;

}

//Сортировка сборок по снижению цены

int y = \_sborki.size();

Sborka z;

for (int i = 1; i < y; i++)

for (int j = i - 1; j >= 0 && \_sborki[j] < \_sborki[j + 1]; j--)

{

z = \_sborki[j]; \_sborki[j] = \_sborki[j + 1]; \_sborki[j + 1] = z;

}

//Удаление одинаковых сборок

z = \_sborki[0];

for (int i = 1; i < \_sborki.size(); i++)

if (z == \_sborki[i])

{

\_sborki.erase(\_sborki.begin() + i);

i--;

}

else z = \_sborki[i];

//Вывод сборок в лист

y = \_sborki.size();

for (int \_i = 0; \_i < y; \_i++) {

\_sborki[\_i].SetName(String::Format("Сборка {0}", \_i + 1));

listBoxConfig->Items->Add(gcnew String(\_sborki[\_i].GetName().c\_str()));

}

listBoxConfig->Enabled = true;

}

Полный список функции формы изменения и сохранения сборки

//Меняем сборку с новым компонентом

public: void SetSborka(Sborka sbor, int type, int selind)

{

\_sborka = sbor;

k = type;

//Выводим все компоненты сборки в лист бокс

textName->Text = (gcnew String(sbor.GetName().c\_str()));

listComponents->Items->Clear();

System::String^ str = gcnew String(\_sborka.GetCard().GetName().c\_str());

str = str + " (" + \_sborka.GetCard().GetCost().ToString() + "р.)";

listComponents->Items->Add(str);

str = gcnew String(\_sborka.GetMother().GetName().c\_str());

str = str + " (" + \_sborka.GetMother().GetCost().ToString() + "р.)";

listComponents->Items->Add(str);

str = gcnew String(\_sborka.GetProts().GetName().c\_str());

str = str + " (" + \_sborka.GetProts().GetCost().ToString() + "р.)";

listComponents->Items->Add(str);

str = gcnew String(\_sborka.GetRam().GetName().c\_str());

str = str + " (" + \_sborka.GetRam().GetCost().ToString() + "р.)";

listComponents->Items->Add(str);

str = gcnew String(\_sborka.GetSata().GetName().c\_str());

str = str + " (" + \_sborka.GetSata().GetCost().ToString() + "р.)";

listComponents->Items->Add(str);

str = gcnew String(\_sborka.GetPower().GetName().c\_str());

str = str + " (" + \_sborka.GetPower().GetCost().ToString() + "р.)";

listComponents->Items->Add(str);

}

void ClearLists()

{

listComponents->Items->Clear();

listAvailable->Items->Clear();

}

//Начата выборка компонента

private: System::Void listComponents\_DoubleClick(System::Object^ sender, System::EventArgs^ e)

{

//Подготавливаем логи для вывода информации о замене

System::String^ str;

listAvailable->Items->Clear();

int \_selected = listComponents->SelectedIndex;

if (\_selected == -1) return;

else if (\_selected == 0) //Если выбрали видеокарту

{

int y = \_cardss.size();

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

{

str = gcnew String((\_cardss)[i].GetName().c\_str());

str = str + " (" + (\_cardss)[i].GetCost().ToString() + "р.)";

listAvailable->Items->Add(str);

}

}

else if (\_selected == 1) //Если выбрали материнскую плату

{

int y = \_motherss.size();

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

{

str = gcnew String(\_motherss[i].GetName().c\_str());

str = str + " (" + \_motherss[i].GetCost().ToString() + "р.)";

listAvailable->Items->Add(str);

}

}

else if (\_selected == 2) //Если выбрали процессор

{

int y = \_processorss.size();

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

{

str = gcnew String(\_processorss[i].GetName().c\_str());

str = str + " (" + \_processorss[i].GetCost().ToString() + "р.)";

listAvailable->Items->Add(str);

}

}

else if (\_selected == 3) //Если выбрали оперативную память

{

int y = \_ramss.size();

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

{

str = gcnew String(\_ramss[i].GetName().c\_str());

str = str + " (" + \_ramss[i].GetCost().ToString() + "р.)";

listAvailable->Items->Add(str);

}

}

else if (\_selected == 4) //Если выбрали жесткий диск

{

int y = \_satss.size();

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

{

str = gcnew String(\_satss[i].GetName().c\_str());

str = str + " (" + \_satss[i].GetCost().ToString() + "р.)";

listAvailable->Items->Add(str);

}

}

else if (\_selected == 5) //Если выбрали блок питания

{

int y = \_powerss.size();

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

{

str = gcnew String(\_powerss[i].GetName().c\_str());

str = str + " (" + \_powerss[i].GetCost().ToString() + "р.)";

listAvailable->Items->Add(str);

}

}

comsel = \_selected;

}

//Начата замена компонента

private: System::Void listAvailable\_DoubleClick(System::Object^ sender, System::EventArgs^ e) {

listNote->Items->Clear();

System::String^ str;

int componentselect = listAvailable->SelectedIndex;

if (componentselect == -1) return;

//Заменили выбранный компонент в сборке

if (comsel == 0) \_sborka.SetConfig(\_cardss[componentselect], \_sborka.GetMother(), \_sborka.GetProts(), \_sborka.GetRam(), \_sborka.GetSata(), \_sborka.GetPower());

else if (comsel == 1) \_sborka.SetConfig(\_sborka.GetCard(), \_motherss[componentselect], \_sborka.GetProts(), \_sborka.GetRam(), \_sborka.GetSata(), \_sborka.GetPower());

else if (comsel == 2) \_sborka.SetConfig(\_sborka.GetCard(), \_sborka.GetMother(), \_processorss[componentselect], \_sborka.GetRam(), \_sborka.GetSata(), \_sborka.GetPower());

else if (comsel == 3) \_sborka.SetConfig(\_sborka.GetCard(), \_sborka.GetMother(), \_sborka.GetProts(), \_ramss[componentselect], \_sborka.GetSata(), \_sborka.GetPower());

else if (comsel == 4) \_sborka.SetConfig(\_sborka.GetCard(), \_sborka.GetMother(), \_sborka.GetProts(), \_sborka.GetRam(), \_satss[componentselect], \_sborka.GetPower());

else if (comsel == 5) \_sborka.SetConfig(\_sborka.GetCard(), \_sborka.GetMother(), \_sborka.GetProts(), \_sborka.GetRam(), \_sborka.GetSata(), \_powerss[componentselect]);

SetSborka(\_sborka, k,0);

buttonClose->Enabled = true;

//Смотрим, совместим ли выбранный компонент в нашей сборке

//Проверка сокета процессора и материнской платы

if (\_sborka.GetMother().GetSocket() != \_sborka.GetProts().GetSocket())

{

listNote->Items->Add("Невозможная сборка - разница в сокетах!");

str = gcnew String(\_sborka.GetMother().GetName().c\_str());

str = str + " использует " + \_sborka.GetMother().GetSocket().ToString() + " сокет.";

listNote->Items->Add(str);

str = gcnew String(\_sborka.GetProts().GetName().c\_str()) + " использует " + \_sborka.GetProts().GetSocket().ToString() + " сокет.";;

listNote->Items->Add(str);

buttonClose->Enabled = false;

}

//Проверка баланса мощности процессора и видеокарты

if (abs(\_sborka.GetCard().GetPoints() - \_sborka.GetProts().GetPoints()) > 30)

{

listNote->Items->Add("Не оптимальная сборка - нарушен баланс видеокарты и процессора.");

if (\_sborka.GetCard().GetPoints() > \_sborka.GetProts().GetPoints()) listNote->Items->Add("Слишком мощная видеокарта.");

else listNote->Items->Add("Слишком мощный процессор.");

}

//Проверка энергопторебления сборки

if ((\_sborka.GetCard().GetTdp() + \_sborka.GetProts().GetTdp() + 175) > \_sborka.GetPower().GetWatt())

{

int tdp = 0;

listNote->Items->Add("Невозможная сборка - слабый блок питания");

str = gcnew String(\_sborka.GetPower().GetName().c\_str());

str = str + " формирует напряжение в " + \_sborka.GetPower().GetWatt().ToString() + " ватт.";

listNote->Items->Add(str);

str = gcnew String(\_sborka.GetCard().GetName().c\_str()) + " потребляет " + \_sborka.GetCard().GetTdp().ToString() + " ватт.";;

listNote->Items->Add(str);

tdp += \_sborka.GetCard().GetTdp();

str = gcnew String(\_sborka.GetProts().GetName().c\_str()) + " потребляет " + \_sborka.GetProts().GetTdp().ToString() + " ватт.";;

listNote->Items->Add(str);

tdp += \_sborka.GetProts().GetTdp();

listNote->Items->Add("Нужды остальной системы - 175 ватт.");

tdp += 175;

str = "Всего система использует " + tdp + " ватт.";

listNote->Items->Add(str);

buttonClose->Enabled = false;

}

//Стоимость сборки

str = "Стоимость сборки составит " + \_sborka.GetCost() + " руб.";

listNote->Items->Add(str);

}

//Закртыие формы

private: System::Void EditForm\_FormClosing(System::Object^ sender, System::Windows::Forms::FormClosingEventArgs^ e) {

e->Cancel = true;

myEvent2(this, e);

this->Hide();

}

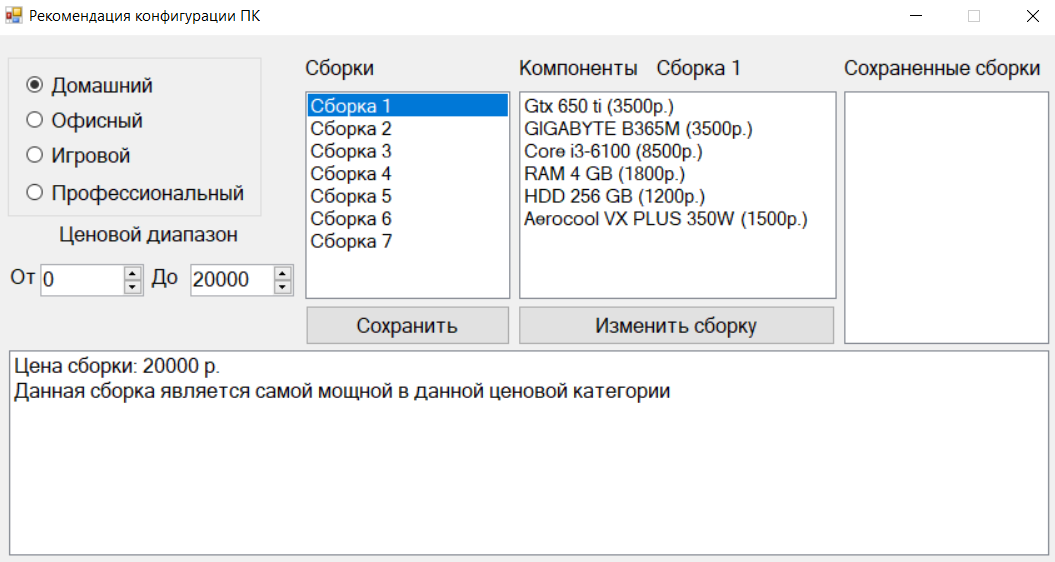
Исходный код проекта написан на c++ с помощью фрэймворка .NET 4.7.2 и доступен в GitHub <https://github.com/Danilych/configurator>

# 5. Этап тестирования

## Оценка точности работы

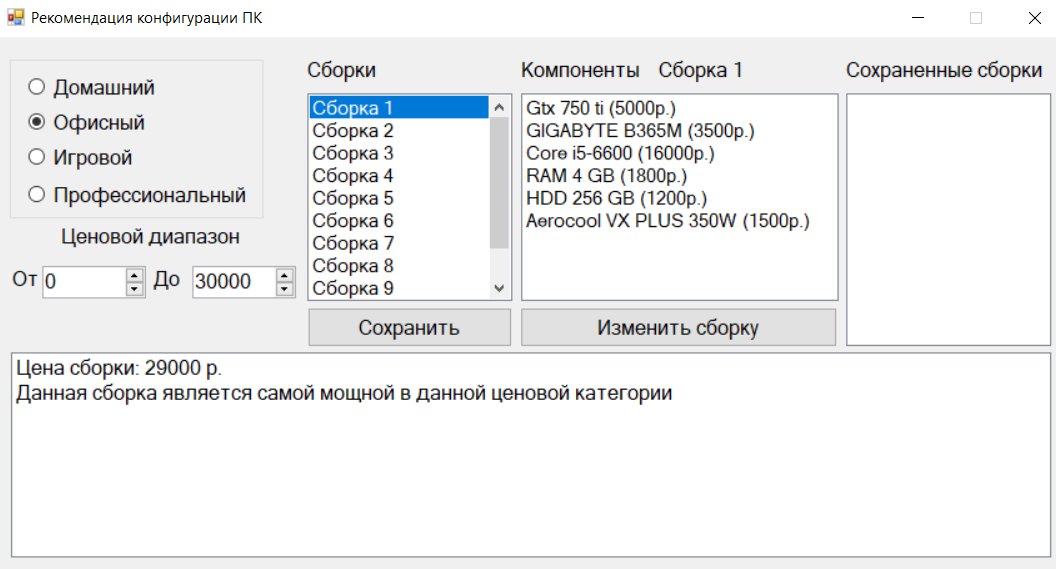
Для оценки экспертной системы мы сделали четыре запроса в разных ценовых диапазонах и посмотрели что нам выдаст программа

Запрос и результат программы:



Наше мнение:  
Сборка является достойной, программа постаралась выбрать хоть что-то по такому низкому бюджету и сумела обеспечить пользователя процессором начального уровня, хоть какой-то видеокартой и оперативной памятью.

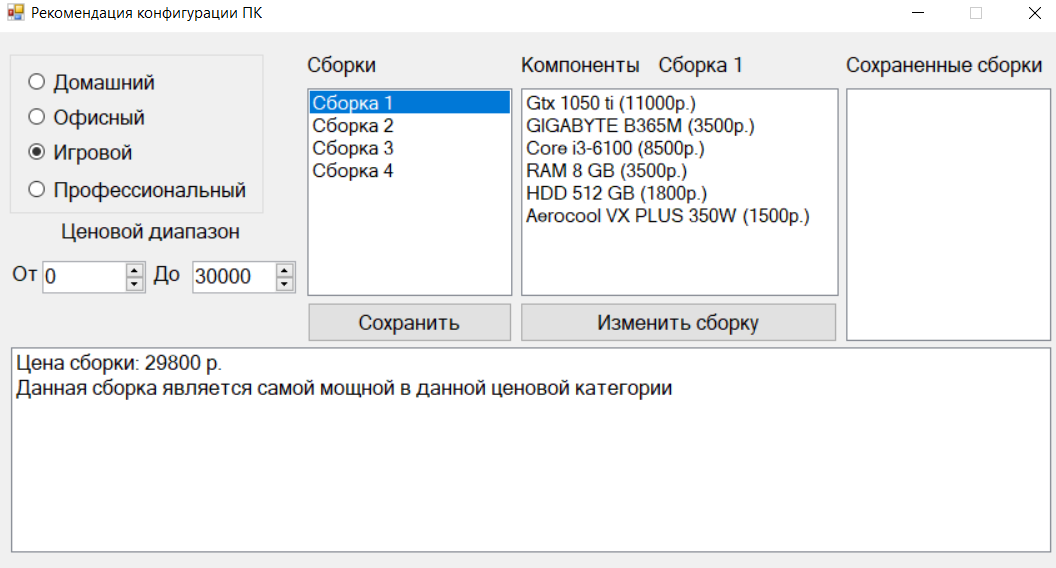
Запрос и результат программы:



Наше мнение:

Не плохая офисная сборка с бюджетом до 30 тысяч рублей. Мощный процессор обеспечивает надежную работу всех офисных приложений, а оперативная память позволит сделать работу более продуктивной.

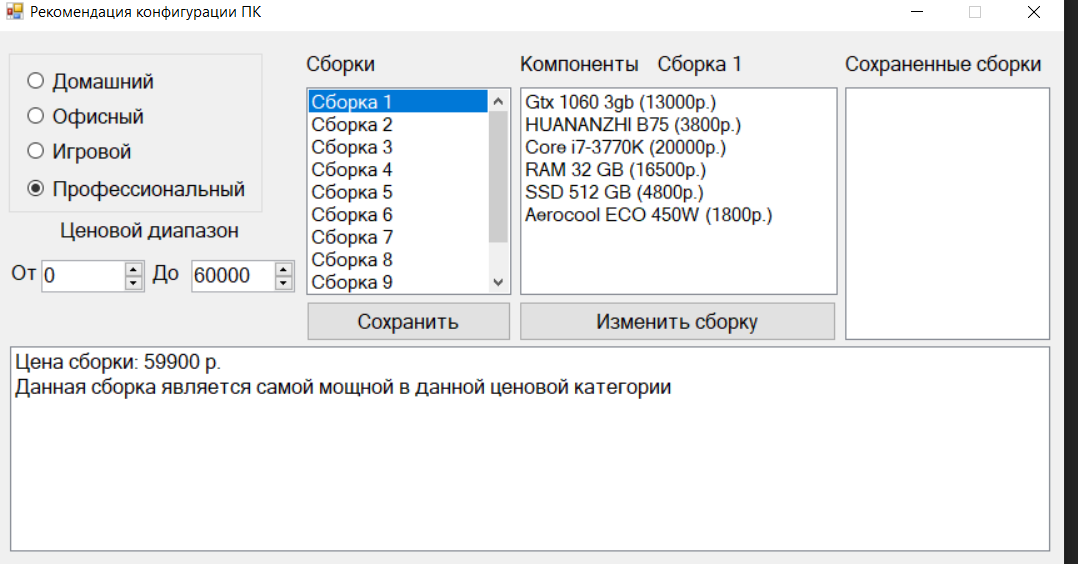
Запрос и результат программы:



Наше мнение:

Достаточно мощная система за такую цену, способна тянуть многие игры на минимальных настройках.

Запрос и результат программы:



Наше мнение:

Данная сборка обеспечивает очень высокую продуктивность. Мощная видеокарта в купе с мощным процессором дают пользователю высокий вычислительный потенциал.

## Вывод

Наша программа справляется со своей задачей более, чем не плохо, единственным недостатком можно назвать малость нашей базы данных. Интерфейс программы понятный, поле внизу даёт пользователю дополнительную информацию о конкретной сборке. Опытный пользователь может сам изменить свою систему Программа подходит широкому кругу пользователей..