МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра РАПС

КУРСОВАЯ РАБОТА

по дисциплине «Программирование и основы алгоритмизации»

Тема: Список автомобилей автосалона

Студент гр. 3403 _____ Задорожный К.Ю

Преподаватель _____ Калинин А.В.

Санкт-Петербург

2024

Содержание

Введение 3
Разработка структуры данных 4
Разработка структуры приложения 5
Создание массива данных 5
Чтение и запись в файл 5
Редактирование 5
Добавление 6
Удаление 6
Табличное представление данных 6
Сортировка по трем полям базового класса 6
Поиск по году производства 7
Графическая визуализация данных 7
Разработка интерфейса 8
Заключение 11
Приложение 12

Введение

Целью данного курсового проекта является получение навыков постановки задачи, алгоритмизации, программирования и отладки приложений на примере решения задачи учета и анализа продукции, услуг в некоторой предметной области.

На основании выбранной тематики курсового проекта необходимо разработать визуальное приложение в программной среде C++ Builder, которое должно поддерживать сопровождение информацией базы данных автосалона в виде текстового файла. Для каждого автомобиля в приложении заносятся данные о его названии, годе выпуска и стоимости в рублях. Также, в зависимости от типа двигателя автомобиля (электрического или внутреннего сгорания), предоставляется возможность внести некоторые дополнительные данные, характерные тому или иному типу автомобиля (тип топлива, объём двигателя и расход топлива в расчете на 100 км – для авто с ДВС; вид электродвигателя, ёмкость батареи и время полной зарядки – для электрокаров). Приложение должно позволять осуществление следующих действий: чтение из текстового файла, запись данных в файл, выбранным пользователем, также добавление, редактирование, удаление, сортировка данных, поиск по некоторым полям для быстроты нахождения нужной информации, графическая интерпретация данных в виде диаграммы.

Разработка структуры данных

Разработка структуры данных приложения основывается на использовании возможностей ООП. Каждой сопровождаемой сущности соответствует объект соответствующего класса. Структура основана на наследовании классов, реализован механизм позднего связывания, для чего определен массив указателей на объекты базового класса, в котором хранятся указатели на объекты дочерних классов.

В приложении предусмотрены следующие типы автомобилей:

- Автомобили с электродвигателем – ELECT
- Автомобили с ДВС - FUEL

Каждому из типов соответствует собственный класс (Electric_Engine, Fuel_Engine). Все они являются производными от базового класса Car, который содержит информацию, актуальную для обоих типов автомобилей: их тип, наименование, год производства и стоимость. Для электроавтомобилей добавляются характеристики электродвигателя, батареи и скорости зарядки, а для топливных автомобилей – описание объема бака, расхода на 100 км и топлива (разновидности). В базовом классе реализована перегрузка конструктора, конструктор копирования, а также виртуальные и чисто виртуальный методы.

Программная реализация классов приведена в далее (см. Приложение).

Разработка структуры приложения

Создание массива данных

В работе использован статический массив *cars[]*, то есть его длина фиксирована уже на этапе компиляции, в который помещены указатели на объекты базового класса. Также создана переменная *cnt*, хранящая количество созданных объектов. Массив и переменная – счетчик объявлены, как глобальные.

```
Car* cars[LEN];
int cnt = 0;
```

Чтение и запись в файл

Первичная функция, которая доступна пользователю – *Read_Data()* это возможность считать данные из текстового файла, который пользователь может выбрать абсолютно сам, за счет компонента *TOpenDialog* (код данной функции представлен в Приложении).

Также после некоторых действий (удаление, редактирование, добавление) пользователя, приложение позволяет сохранить данные изменения в текстовый документ, который пользователь может также выбрать сам с помощью компонента *TOpenDialog – функция Save_Data()* (см. Приложение).

Редактирование

Функция редактирования позволяет непосредственно изменить данные выбранного элемента в таблице. Данная функция может быть вызвана двумя способами: двойным нажатием по элементу в таблице или через пункт контекстного меню *TPopupMenu*. При вызове открывается новая форма, в которую пользователь может занести необходимые изменения.

ShowEditMenu – создает новую форму и заполняет соответствующие поля необходимыми значениями. CreateObject – считывает данные из второй формы редактирования.

Добавление

Добавление в приложение осуществляется также, как и редактирование, двумя путями: через отдельную кнопку «Добавить» или через выбор соответствующего пункта в контекстном меню. Функции, реализующие добавление элемента в массив:

- Show_CreateMenu() – создаёт дополнительную форму и соответствующие задаче создания объекта поля и выпадающие списки.
- void TLinkMainForm::Add_Object() – создает объект класса на основании считанных данных, введенных пользователем в поля меню создания.

Удаление

void TLinkMainForm::Pop_Object(int index) – функция, получающая на вход индекс выбранного для удаления пользователем. Осуществляется через контекстное меню (код см. Приложение).

Табличное представление данных

Компонент TListVeiw позволяет выводит данные в табличном виде. В приложении с помощью этого компонента реализовано отображение данных, полученных из полей объектов классов массива указателей.

Функцией void TLinkMainForm::Show_Data(Car* array[], int n) данные из массива array выводятся на экран пользователя (код см. Приложение).

Сортировка по трем полям базового класса

Реализация логики сортировки следующая: пользователь считывает данные из файла (или заполняет вручную), далее из выпадающего списка выбирает одно из 3х доступных полей сортировки: Название, Год, Стоимость. После щелчка по одному из полей данные таблицы выстраиваются в искомом порядке.

Алгоритм сортировки – Быстрая сортировка. Программная реализация представлена в Приложении.

Поиск по году производства

void TLinkMainForm::GlobalSearch(Car* array[], int x, int* res, int &k) – функция поиска(код см. Приложении). Поиск реализован по полю года производства автомобилей. Алгоритм, выбранный для реализации поиска – бинарный поиск, который требует, чтобы массив был отсортирован заранее. Бинарный поиск заточен на поиск одного необходимого элемента, однако совпадений по году может быть более одного, поэтому данный алгоритм дополнен линейным проходом справа и слева от найденного элемента и вывод их в таблицу для пользователя (см. Приложение).

Графическая визуализация данных

В качестве визуализации данных создана возможность внутри приложения построить круговую диаграмму, отражающую процентное соотношение между автомобилями в автосалоне по году производства. Рисование секторов реализовано с помощью втсроенной функции Pie() (см. Приложение).

Разработка интерфейса

Иллюстрация разработанного пользовательского интерфейса представлена ниже на рисунках 1-6:

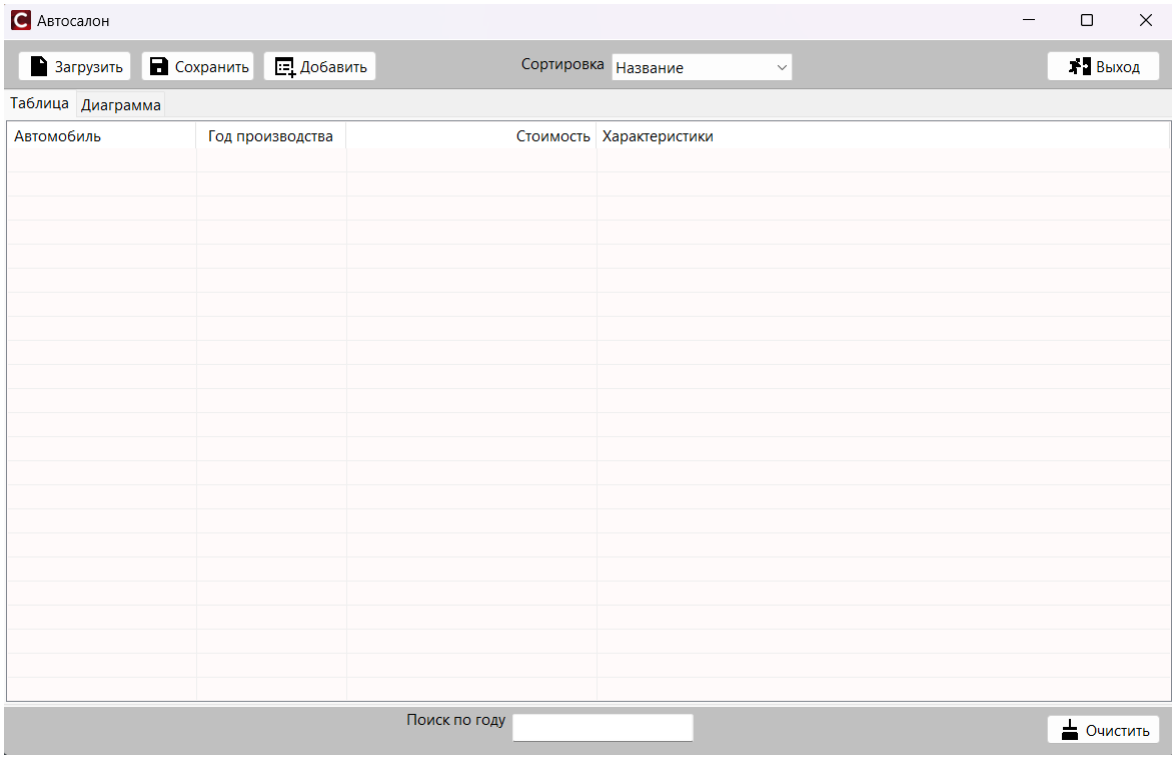

Рисунок 1 – Вид приложения при запуске

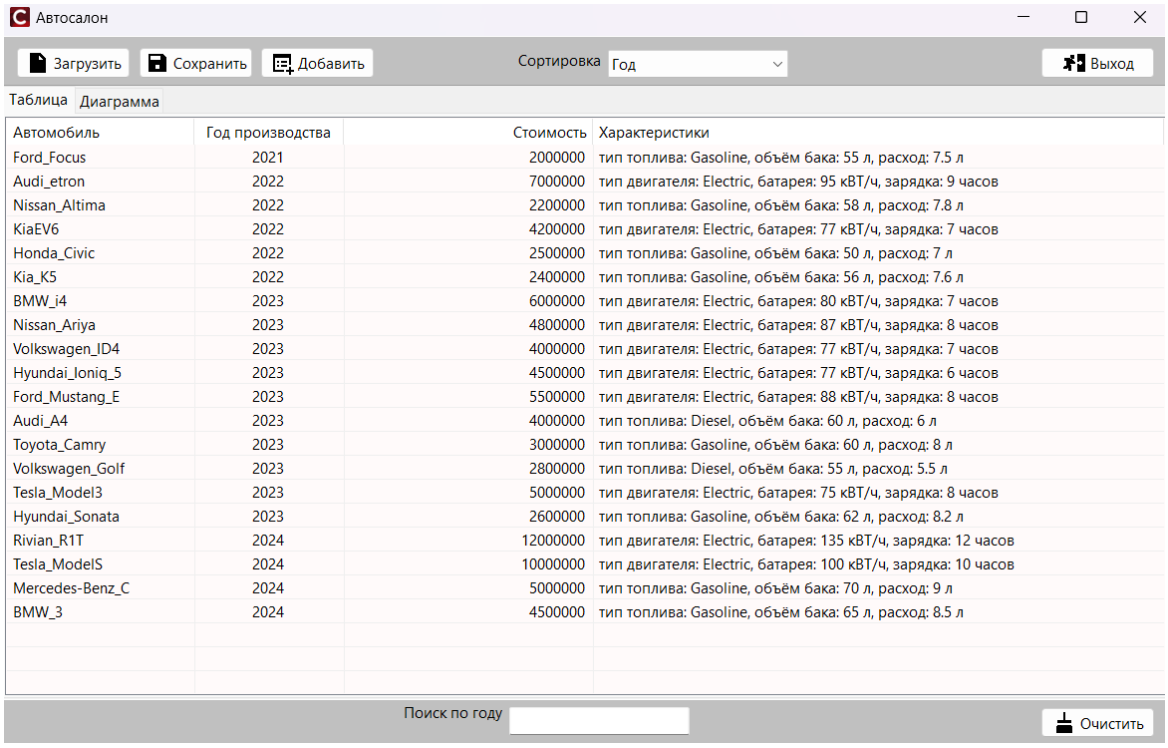

Рисунок 2 – Табличное представление данных после чтения их из файла + сортировка по полю «Год»

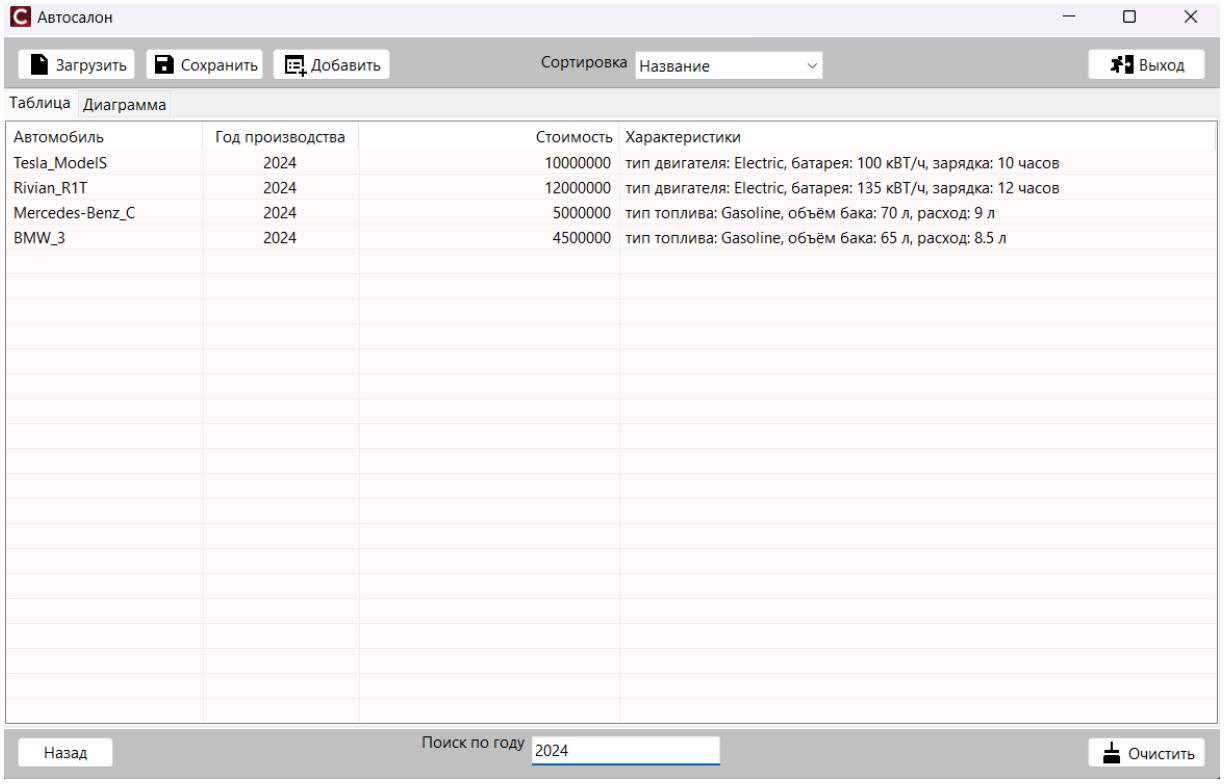

Рисунок 3 – Вывод всех совпадений на экран в результате поиска по году «2024»

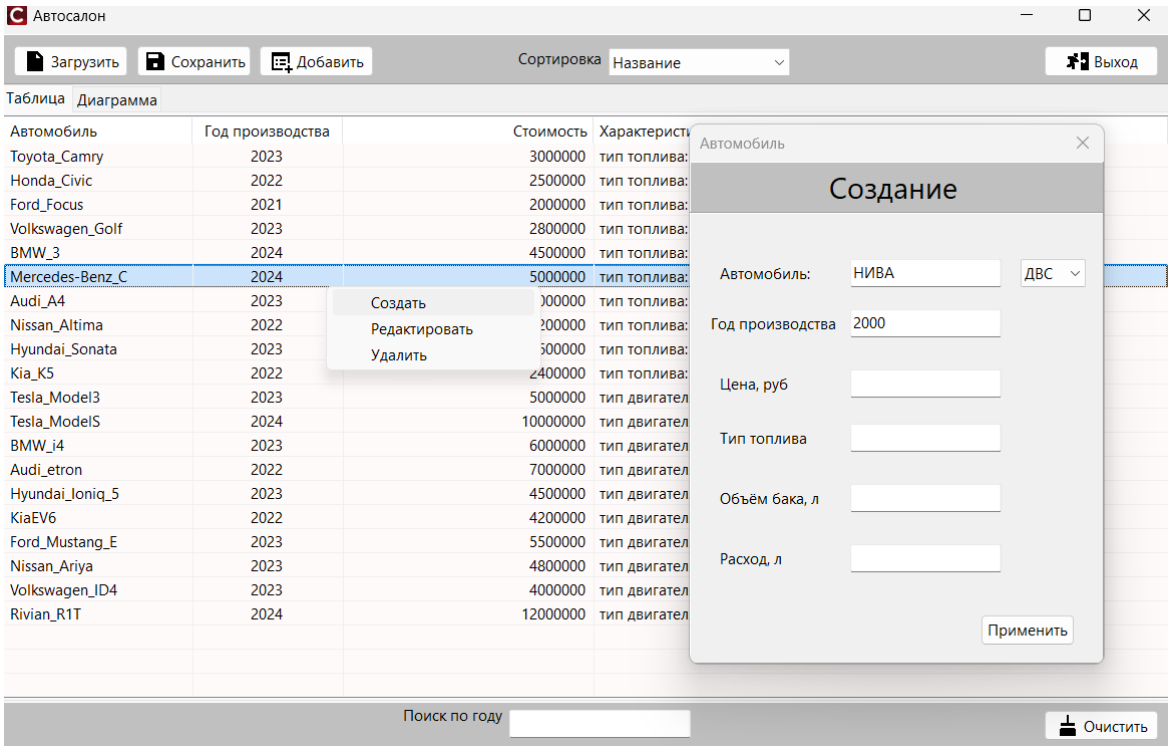

Рисунок 4 – Добавление нового элемента

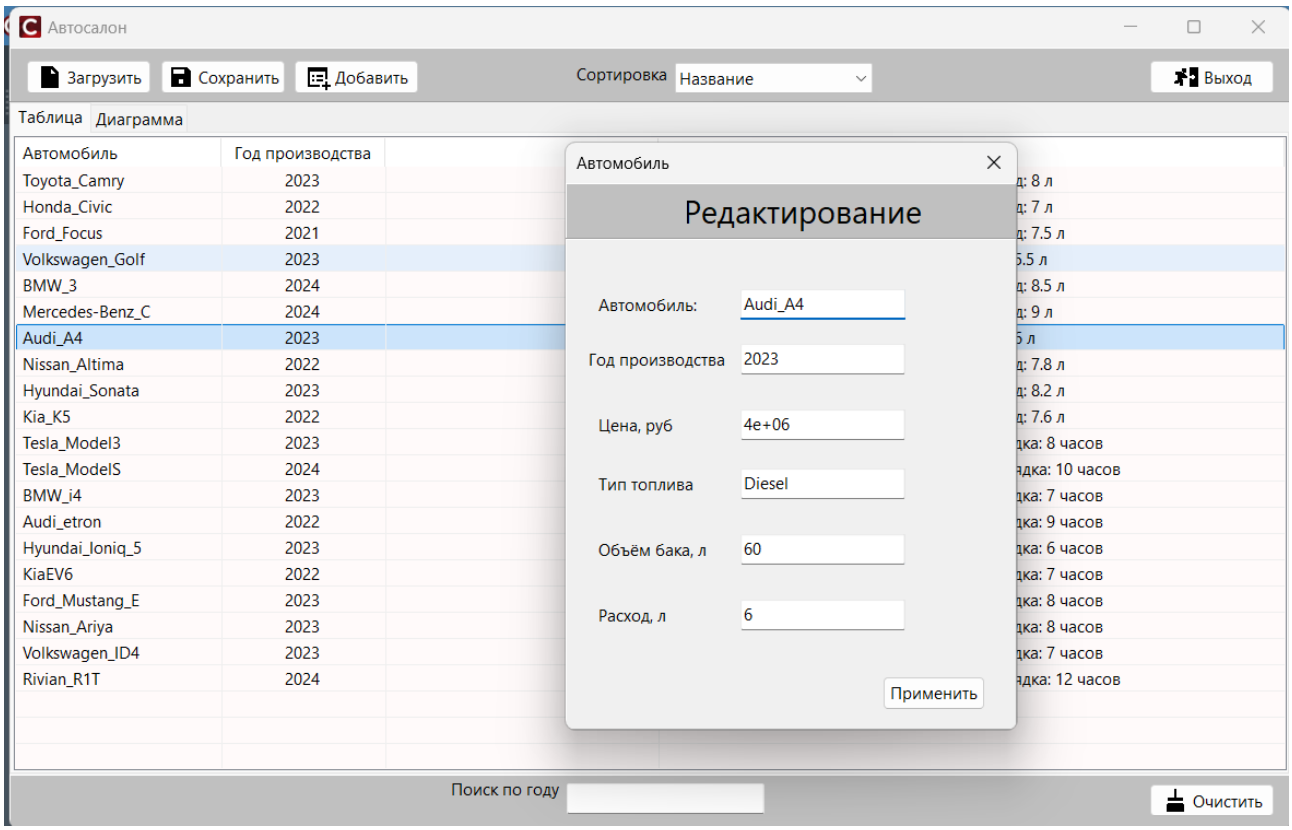

Рисунок 5 – Редактирование выбранного элемента

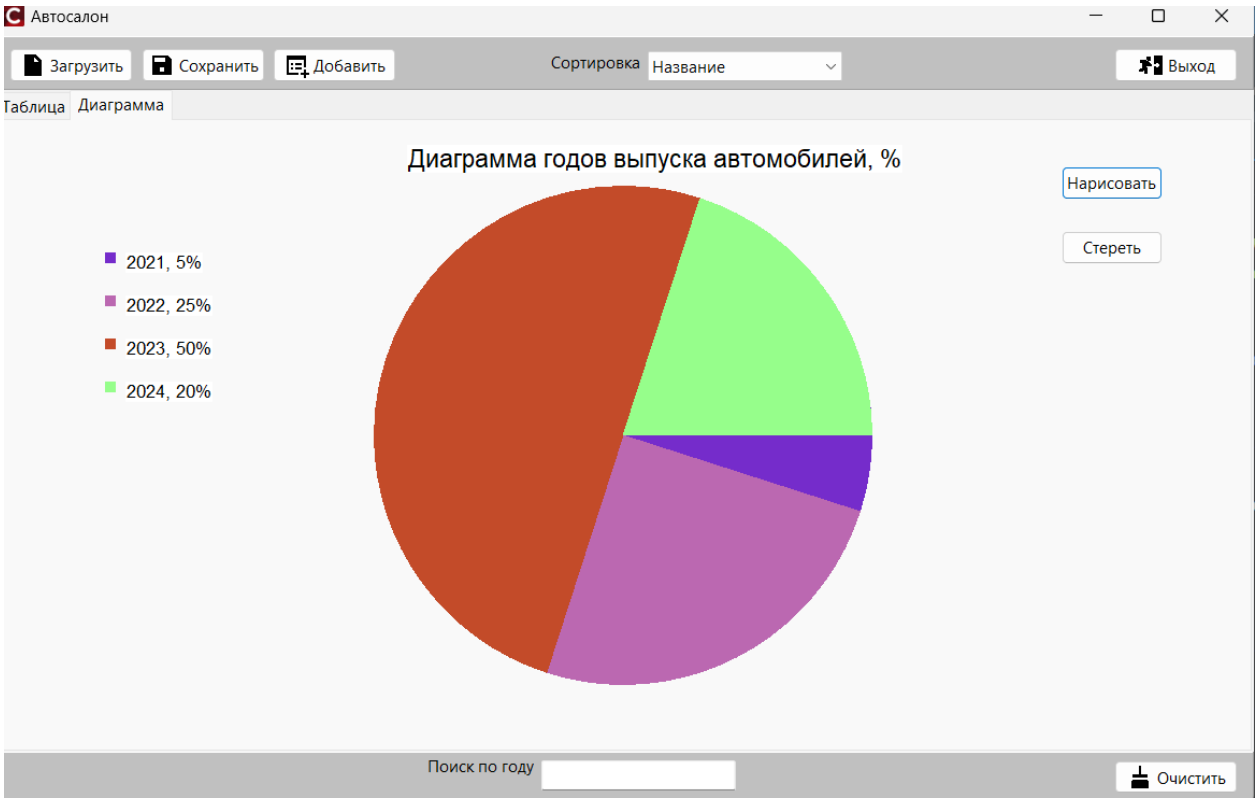

Рисунок 6 – Гарфическое изображение данных

Заключение

В данном курсовом проекте успешно реализовано визуальное приложение позволяющее располагать инструментами для оперирования с данными об автомобилях некоторого автосалона, а именно пользователю дана возможность чтения и записи информации в файл, ее редактирование, удаление и добавление, также приложение поддерживает возможность сортировки и поиска по некоторым из полей.

В ходе работы над проектом были получены навыки с работой в библиотеки VCL C++ Builder, изучены некоторые понятия, связанные с ООП: понятие о классах, наследованием классов, полиморфизмом, поздним связыванием и т.д.

Приложение

Код файла MainForm.h

```
class Car
{
private:
     char      TYPE[LEN];
     char      name[LEN];
     int       year;
     double    price;

public:

     Car(char* Type, char* Name, int Year, double Price);
     Car(AnsiString Type, AnsiString Name, AnsiString Year,
          AnsiString Price);

     Car(const Car& C);

     virtual void GetValues(char* buf) = 0;

     char*     GetName()      { return name; }
     int  GetYear()      { return year; }
     double    GetPrice()     { return price; }

     char* GetTYPE() { return TYPE; }

     virtual char* GetFuel_Type() {return 0;}
     virtual float GetTank_Vol()  {return 0;}
     virtual float GetFuel_Rate() {return 0;}

     virtual char* GetEl_type() {return 0;}
     virtual int   GetBat_cpt() {return 0;}
     virtual int   GetCh_time() {return 0;}
};

class Fuel_Engine : public Car
{
private:
     char      fuel_type[LEN];
     float     tank_volume;
     float     fuel_rate;
public:

     Fuel_Engine(char* Type, char* Name, int Year, double Price,
     char* ftype, float tvol, float frate);

     Fuel_Engine(AnsiString Type, AnsiString Name, AnsiString
Year,
          AnsiString Price, AnsiString ftype, AnsiString tvol,
               AnsiString frate);

     Fuel_Engine(const Fuel_Engine& F);
```

```
     char*     GetFuel_Type()      override {return fuel_type;}
     float     GetTank_Vol()  override {return tank_volume;}
     float     GetFuel_Rate()      override {return fuel_rate;}

     void GetValues(char* buf) override;

};

class Electric_Engine : public Car
{
private:

     char      elengine_type[LEN];
     int  battery_capacity;
     int  charging_time;

public:

     Electric_Engine(char* Type, char* Name, int Year, double
Price,
          char* eltype, int btcpt, int ctime);

     Electric_Engine(AnsiString Type, AnsiString Name, AnsiString
Year,
          AnsiString Price, AnsiString eltype, AnsiString btcpt,
               AnsiString ctime);

     Electric_Engine(const Electric_Engine& E);

     char* GetEl_type() override {return elengine_type; }
     int  GetBat_cpt()   override {return battery_capacity; }
     int  GetCh_time()   override {return charging_time; }

     void GetValues(char* buf) override;

};
Car* cars[LEN];
int cnt = 0;

class TLinkMainForm : public TForm
{
__published:   // IDE-managed Components
     TListView *AutoList;
     TPanel *UpPanel;
     TPanel *DownLinkPanel;
     TEdit *SearchEdit;
     TLabel *SearchLabel;
     TPageControl *MainPages;
     TTabSheet *TableSheet;
     TTabSheet *ViewSheet;
     TPaintBox *pb;
```

```
        TComboBox *SortingCB;
        TLabel *SortL;
        TOpenDialog *FilesDialog;
        TPopupMenu *PopupMenu;
        TMenuItem *Insrert;
        TMenuItem *Change;
        TMenuItem *Pop;
        TBitBtn *BitBtn1;
        TBitBtn *BitBtn2;
        TBitBtn *BitBtn3;
        TBitBtn *BitBtn4;
        TBitBtn *BitBtn5;
        TButton *Button1;
        TButton *Button2;
        TButton *GoBackBt;
        void __fastcall Save(TObject *Sender);
        void __fastcall ReadData(TObject *Sender);
        void __fastcall ExitApp(TObject *Sender);
        void __fastcall ClearAuto(TObject *Sender);
        void __fastcall EditProperties(TObject *Sender);
        void __fastcall SortActivation(TObject *Sender);
        void __fastcall SearchAct(TObject *Sender, System::WideChar
&Key);
        void __fastcall InsrertClick(TObject *Sender);
        void __fastcall PopClick(TObject *Sender);
        void __fastcall pbPaint(TObject *Sender);
        void __fastcall Button2Click(TObject *Sender);
        void __fastcall GoBackBtClick(TObject *Sender);

    private:
        void Read_Data(char* filename);
        void Save_Data(char* filename);
        void QuickSorting(int sort_key, Car* array[]);
        void Clear_Data();
        void Show_Data(Car* array[] = cars, int n = cnt);
        void Show_EditMenu(int index);
        void GlobalSearch(Car* array[], int x, int* res, int &k);
        void Show_CreateMenu();
        void Pop_Object(int index);
        void sort(int key, int L, int R, Car* array[]);
        bool sorting_key(int key, int x, int i, Car* array[]);

        void BubleSort(int sort_key, Car* array[]);

    public:         // User declarations
        __fastcall TLinkMainForm(TComponent* Owner);
        void Updata_Object();
        void Add_Object();
    };
    extern PACKAGE TLinkMainForm *LinkMainForm;
```

Код файла MainForm.cpp

```
#define NAME_SKEY 0
#define YEAR_SKEY 1
#define PRICE_SKEY 2
#include <cmath>
using namespace std;

TLinkMainForm *LinkMainForm;

__fastcall TLinkMainForm::TLinkMainForm(TComponent* Owner)
     : TForm(Owner)
{
     SortingCB->ItemIndex = 0;
}

void TLinkMainForm::Read_Data(char* filename)
{
     FILE *inp = fopen(filename,"r");

     if(!inp)
     {
          ShowMessage("Ошибка открытия файла!");
          return;
     }

     char      type[LEN];
     char      name[LEN];
     int  year;
     double    price;

     while(!feof(inp))
     {
          fscanf(inp, "%s", type);

          if ( !memcmp(type, "FUEL", 4) )
          {
               char      ftype[LEN];
               float     tvol;
               float     frate;

               if( fscanf(inp, "%s %d %lf %s %f %f\n", name, &year,
               &price,
                    ftype, &tvol, &frate) )
               {
                    cars[cnt] = new Fuel_Engine(type, name, year,
                    price,
                         ftype, tvol, frate);
               }
               else
               {
                    ShowMessage(cars[cnt]->GetTYPE());
```

```
                    ShowMessage("Ошибка в чтении данных!");
                }
            }

            else if ( !memcmp(type, "ELECT", 5) )
            {
                char      eltype[LEN];
                int  btcpt;
                int  ctime;

                if (fscanf(inp, "%s %d %lf %s %d %d\n", name, &year,
                &price,
                    eltype, &btcpt, &ctime) )
                {
                    cars[cnt] = new Electric_Engine(type, name,
                year, price,
                        eltype, btcpt, ctime);
                }
                else
                {
                    ShowMessage(cars[cnt]->GetTYPE());
                    ShowMessage("Ошибка в чтении данных!");
                }
            }
            cnt++;
        }

        fclose(inp);

        ShowMessage("Данные прочитаны успешно!");
        Show_Data();
}

void TLinkMainForm::Updata_Object()
{
        AnsiString index_str = DetailedForm->IndexL->Caption;
        AnsiString N = DetailedForm->NameEd->Text;
        AnsiString Y = DetailedForm->YearEd->Text;
        AnsiString P = DetailedForm->PriceEd->Text;

        AnsiString A1 = DetailedForm->AddEd1->Text;
        AnsiString A2 = DetailedForm->AddEd2->Text;
        AnsiString A3 = DetailedForm->AddEd3->Text;

        int index = atoi(index_str.c_str());

        AnsiString T;

        T = AnsiString(cars[index]->GetTYPE());

        if ( !memcmp(cars[index]->GetTYPE(), "FUEL", 4) )
            {
```

```
                Car* buf =  new Fuel_Engine(T, N, Y, P, A1, A2, A3);

                cars[index] = buf;

                delete [] buf;

           }

      else if ( !memcmp(cars[index]->GetTYPE(), "ELECT", 5) )
           {
                Car* buf = new Electric_Engine(T, N, Y, P, A1, A2,
                A3);

                 cars[index] = buf;

                delete [] buf;
           }

      Show_Data();

      ShowMessage("Данные изменены!");

}

void TLinkMainForm::Add_Object()
{
      AnsiString N = DetailedForm->NameEd->Text;
      AnsiString Y = DetailedForm->YearEd->Text;
      AnsiString P = DetailedForm->PriceEd->Text;

      AnsiString A1 = DetailedForm->AddEd1->Text;
      AnsiString A2 = DetailedForm->AddEd2->Text;
      AnsiString A3 = DetailedForm->AddEd3->Text;

      AnsiString T = (!DetailedForm->TypeCb->ItemIndex) ? "FUEL" :
      "ELECT";


      if ( !memcmp(T.c_str(), "FUEL", 4) )
           {
                cars[cnt] = new Fuel_Engine(T, N, Y, P, A1, A2, A3);
           }

      else if ( !memcmp(T.c_str(), "ELECT", 5) )
           {
                cars[cnt] = new Fuel_Engine(T, N, Y, P, A1, A2, A3);
           }
      cnt++;
      Show_Data();
      ShowMessage("Данные добавлены!");
}
```

```
void TLinkMainForm::Show_Data(Car* array[], int n)
{
     AutoList->Items->BeginUpdate();
     AutoList->Items->Clear();

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

          TListItem* item = AutoList->Items->Add();

          item->Caption = array[i]->GetName();

          item->Data = (void*)i;

          int dataValue = (int)item->Data; // запоминаем индекс

          item->SubItems->Add(array[i]->GetYear());
          item->SubItems->Add(array[i]->GetPrice());

          char buf[100];

          if ( !memcmp(array[i]->GetTYPE(), "FUEL", 4) )
          {
               sprintf(buf, "тип топлива: %s, объём бака: %g л,
расход: %g л",
                    array[i]->GetFuel_Type(), array[i]-
>GetTank_Vol(),
                         array[i]->GetFuel_Rate() );

               item->SubItems->Add(buf);
          }

          else if ( !memcmp(array[i]->GetTYPE(), "ELECT", 5) )
          {
               sprintf(buf, "тип двигателя: %s, батарея: %d кВТ/ч,
               зарядка: %d часов",
                    array[i]->GetEl_type(), array[i]->GetBat_cpt(),
                         array[i]->GetCh_time());

               item->SubItems->Add(buf);
          }
     }

     AutoList->Items->EndUpdate();
}

void TLinkMainForm::Show_EditMenu(int index)
{
     DetailedForm->Head1L->Visible = true;
     DetailedForm->Head2L->Visible = false;

     DetailedForm->TypeCb->Visible = false;
```

```
        DetailedForm->AddEd1->Enabled = true;
        DetailedForm->AddEd2->Enabled = true;
        DetailedForm->AddEd3->Enabled = true;

        if ( !memcmp(cars[index]->GetTYPE(), "FUEL", 4) )
        {
             DetailedForm->AddL1->Caption = "Тип топлива" ;
             DetailedForm->AddL2->Caption = "Объём бака, л" ;
             DetailedForm->AddL3->Caption = "Расход, л";
        }

        else if ( !memcmp(cars[index]->GetTYPE(), "ELECT", 5) )
        {
             DetailedForm->AddL1->Caption = "Тип двигателя" ;
             DetailedForm->AddL2->Caption = "Батарея, кВт/ч" ;
             DetailedForm->AddL3->Caption = "Зарядка, ч";
        }

        char*     properties = new char[100];

        char      name[LEN];
        char      year[LEN];
        char price[LEN];
        char add1[LEN];
        char      add2[LEN];
        char      add3[LEN];

        cars[index]->GetValues(properties);

        sscanf(properties, "%[^,],%[^,],%[^,],%[^,],%[^,],%[^,]",
        name, year, price, add1, add2, add3);

        DetailedForm->NameEd->Text = name;
        DetailedForm->YearEd->Text = year;
        DetailedForm->PriceEd->Text = price;
        DetailedForm->AddEd1->Text = add1;
        DetailedForm->AddEd2->Text = add2;
        DetailedForm->AddEd3->Text = add3;

        delete[] properties;
}

void TLinkMainForm::Save_Data(char* filename)
{
        FILE *inp = fopen(filename,"w");

        if(!inp)
        {
             ShowMessage("Ошибка открытия файла!");
             return;
        }
```

```
     for (int i = 0; i < cnt; i++)
     {
          fprintf(inp, "%s %s %d %lf ", cars[i]->GetTYPE(),
     cars[i]->GetName(), cars[i]->GetYear(), cars[i]->GetPrice());

          if (!memcmp(cars[i]->GetTYPE(), "FUEL", 4))

               fprintf(inp, "%s %f %f\n", cars[i]->GetFuel_Type(),
     cars[i]->GetTank_Vol(), cars[i]->GetFuel_Rate());

          else if (!memcmp(cars[i]->GetTYPE(), "ELECT", 5))

               fprintf(inp, "%s %d %d\n", cars[i]->GetEl_type(),
                    cars[i]->GetBat_cpt(), cars[i]->GetCh_time());
     }

     fclose(inp);

     ShowMessage("Данные успешно записаны!");
}

void TLinkMainForm::QuickSorting(int sort_key, Car* array[])
{
     sort(sort_key, 0, cnt-1, array);
}

void TLinkMainForm::sort(int key, int L, int R, Car* array[])
{
     if (L >= R) return;


     int x = min( {L, R, (L + R) / 2});

     int i = L;
     int j = R;

     while(i <= j)
     {
          while(sorting_key(key, i, x, array)) i++;
          while(sorting_key(key, x, j, array)) j--;

          if (i <= j)
          {
               Car* buf =     array[i];
               array[i]  = array[j];
               array[j]  =    buf;

               i++; j--;
          }
     }

     if(L < j) sort(key, L, j, array);
```

```
     if(i < R) sort(key, i, R, array);
}

void TLinkMainForm::BubleSort(int sort_key, Car* array[])
{
     for(int i = 1; i < cnt; i++)
          for (int j = cnt - 1; j >= i; j--)
               if (sorting_key(sort_key, j, j-1, array))
               {
                    Car* buf = array[j-1];
                    array[j-1] = array[j];
                    array[j] = buf;
               }
}

void TLinkMainForm::Clear_Data()
{
     for (int i = 0; i < cnt; i++)
     {
          delete cars[i];
          cars[i] = NULL;
     }

     cnt = 0;
     Show_Data();
     ShowMessage("Данные очищены!");
}

void TLinkMainForm::GlobalSearch(Car* array[], int x, int* res,
int &k)
{

     int L = 0, R = cnt - 1;

     while(L <= R)
     {
          int a = (L+R)/2;

          if (array[a]->GetYear() < x)
               L = a+1;
          else if(array[a]->GetYear() > x)
                R = a-1;
          else
          {
               res[k] = a;
               k++;
               break;
          }
     }

     int l = res[0] - 1;
     while (l >= 0 && array[l]->GetYear() == x) {
```

```
            res[k] = l;
            k++;
            l--;
        }

        int r = res[0] + 1;
        while (r < cnt && array[r]->GetYear() == x) {
            res[k] = r;
            k++;
            r++;
        }
}

void TLinkMainForm::Show_CreateMenu()
{
        DetailedForm->Head1L->Visible = false;
        DetailedForm->Head2L->Visible = true;

        DetailedForm->TypeCb->Visible = true;

        DetailedForm->NameEd->Text    = "";
        DetailedForm->PriceEd->Text = "";
        DetailedForm->YearEd->Text    = "";

        DetailedForm->AddL1->Caption = "н/о";
        DetailedForm->AddEd1->Text = "";
        DetailedForm->AddEd1->Enabled = false;

        DetailedForm->AddL2->Caption = "н/о";
        DetailedForm->AddEd2->Text = "";
        DetailedForm->AddEd2->Enabled = false;

        DetailedForm->AddL3->Caption = "н/о";
        DetailedForm->AddEd3->Text = "";
        DetailedForm->AddEd3->Enabled = false;
}

void TLinkMainForm::Pop_Object(int index)
{
            delete cars[index];

            for (int i = index; i < cnt - 1; i++)
            {
                cars[i] = cars[i + 1];
            }

            cnt--;
            Show_Data();
}

bool TLinkMainForm::sorting_key(int key, int x, int i, Car*
array[]) {
```

```
     switch (key)
     {
     case NAME_SKEY:
          return strcmp(array[x]->GetName(), array[i]->GetName()) <
0;
          break;

     case YEAR_SKEY:
          return array[x]->GetYear() < array[i]->GetYear();
          break;

     case PRICE_SKEY:
          return array[x]->GetPrice() < array[i]->GetPrice();
          break;

    default:
          return false;
     }
}
//---------------------------------------------------------------------
void __fastcall TLinkMainForm::GoBackBtClick(TObject *Sender)
{
     GoBackBt->Visible = false;
     SearchEdit->Text = "";
     Show_Data();
}

void __fastcall TLinkMainForm::Save(TObject *Sender)
{
     char filename[MAX_PATH];

     if (FilesDialog->Execute())
     {
          StrPCopy(filename, FilesDialog->FileName.c_str());
          Save_Data(filename);
     }
}

void __fastcall TLinkMainForm::ReadData(TObject *Sender)
{
     char filename[MAX_PATH];

     if (FilesDialog->Execute())
     {
          StrPCopy(filename, FilesDialog->FileName.c_str());
          Read_Data(filename);
     }
}

void __fastcall TLinkMainForm::ExitApp(TObject *Sender)
{
```

```
     Close();
}

void __fastcall TLinkMainForm::ClearAuto(TObject *Sender)
{
     Clear_Data();
}

void __fastcall TLinkMainForm::EditProperties(TObject *Sender)
{

     if(AutoList->Selected == nullptr || cnt == 0) { return; }

     DetailedForm->Visible = true;

     int index = (int)((TListItem*)AutoList->Selected)->Data;

     Show_EditMenu(index);

     DetailedForm->IndexL->Visible = false;

     DetailedForm->IndexL->Caption = index;
}

void __fastcall TLinkMainForm::SortActivation(TObject *Sender)
{
     int sort_key = SortingCB->ItemIndex;

     QuickSorting(sort_key, cars);
     Show_Data();
}

void __fastcall TLinkMainForm::SearchAct(TObject *Sender,
System::WideChar &Key)
{
     if (Key != VK_RETURN) { return; }

     char buf[LEN];
     AnsiString temp;

     temp = SearchEdit->Text;
     int x = atoi(temp.c_str());

     Car* BUF[LEN], *RES[LEN];

     for(int i = 0; i < cnt; i++) {BUF[i] = cars[i];}

     int res[LEN], k = 0;

     QuickSorting(YEAR_SKEY, BUF);

     GlobalSearch(BUF, x, res, k);
```

```
     for (int i = 0; i < k; i++) { RES[i] = BUF[res[i]];}

   Show_Data(RES, k);

   GoBackBt->Visible = true;

   Key = 0;
}

void __fastcall TLinkMainForm::InsrertClick(TObject *Sender)
{
     if(cnt == LEN)
     {
          ShowMessage("Массив переполнен!");
          return;
     }

     DetailedForm->Visible = true;

     Show_CreateMenu();
}

void __fastcall TLinkMainForm::PopClick(TObject *Sender)
{
     if(AutoList->Selected == nullptr || cnt == 0) { return; }

     int index = (int)(AutoList->Selected)->Data; //индекс

     Pop_Object(index);
}


//constructors
Car::Car(char* Type, char* Name, int Year, double Price)
{
     strcpy(TYPE, Type);
     strcpy(name, Name);
     year = Year;
     price = Price;
}

Car::Car(AnsiString Type, AnsiString Name, AnsiString Year,
     AnsiString Price)
{
     strcpy(TYPE, Type.c_str());
     strcpy(name, Name.c_str());
     year = atoi(Year.c_str());
     price = atof(Price.c_str());
}

Car::Car(const Car& C)
```

```
{
     strcpy(TYPE, C.TYPE);
     strcpy(name, C.name);
     year = C.year;
     price = C.price;
}

Fuel_Engine::Fuel_Engine(char* Type, char* name, int year, double
price,
     char* ftype, float tvol, float frate) : Car(Type, name, year,
price)
{
      strcpy(fuel_type, ftype);
      tank_volume = tvol;
      fuel_rate = frate;
}

Fuel_Engine::Fuel_Engine(AnsiString Type, AnsiString Name,
AnsiString Year,
          AnsiString Price, AnsiString ftype, AnsiString tvol,
               AnsiString frate) : Car(Type, Name, Year, Price)
{
     strcpy(fuel_type, ftype.c_str());
     tank_volume = atof(tvol.c_str());
     fuel_rate = atof(frate.c_str());
}

Fuel_Engine::Fuel_Engine(const Fuel_Engine& F) : Car(F)
{
     strcpy(fuel_type, F.fuel_type);
     tank_volume = F.tank_volume;
     fuel_rate = F.fuel_rate;
}

Electric_Engine::Electric_Engine(char* Type, char* name, int year,
     double price, char* eltype, int btcpt, int ctime) :
          Car(Type, name, year, price)
{
     strcpy(elengine_type, eltype);
     battery_capacity = btcpt;
     charging_time = ctime;
}

Electric_Engine::Electric_Engine(AnsiString Type, AnsiString Name,
AnsiString Year,
          AnsiString Price, AnsiString eltype, AnsiString btcpt,
               AnsiString ctime) : Car(Type, Name, Year, Price)
{
     strcpy(elengine_type, eltype.c_str());
     battery_capacity = atoi(btcpt.c_str());
     charging_time = atoi(ctime.c_str());
}
```

```
Electric_Engine::Electric_Engine(const Electric_Engine& E) :
Car(E)
{
     strcpy(elengine_type, E.elengine_type);
     battery_capacity = E.battery_capacity;
     charging_time = E.charging_time;
}
//class methods

void Fuel_Engine::GetValues(char* buf)
{
     sprintf(buf, "%s,%d,%lg,%s,%g,%g", GetName(), GetYear(),
GetPrice(),
          GetFuel_Type(), GetTank_Vol(), GetFuel_Rate());
}

void Electric_Engine::GetValues(char* buf)
{
     sprintf(buf, "%s,%d,%lg,%s,%d,%d", GetName(), GetYear(),
GetPrice(),
          GetEl_type(), GetBat_cpt(), GetCh_time());
}

void __fastcall TLinkMainForm::pbPaint(TObject *Sender)
{
     if (!cnt) return;
     float w = pb->ClientWidth;
     float h = pb->ClientHeight;
     float x0 = w / 2;
     float y0 = h / 2;
     float r = min(x0, y0) * 0.8;

     double total = 0;
     for(int i = 0; i < cnt; i++)
     {
          total += cars[i]->GetYear();
     }

     Car* temp[LEN];
     for (int i = 0; i < cnt; i++) { temp[i] = cars[i];}

     QuickSorting(YEAR_SKEY, temp);

     int data[cnt][2];
     int ind = 0, i = 0;

     while (i < cnt)
     {
          int k = 1;
          int year = temp[i]->GetYear();
```

```
            int j = i + 1;
            while (j < cnt && temp[j]->GetYear() == year)
            {
                  k++;
                  j++;
            }

            data[ind][0] = year;
            data[ind][1] = k;
            ind++;
            i = j;
      }

      double angl0 = 0;
      i = 0;

      double x1 = x0 - r;
      double y1 = y0 - r;
      double x2 = x0 + r;
      double y2 = y0 + r;

      double x3, x3_prev =  0;
      double y3, y3_prev =  0;

      double x4 = x2;
      double y4 = 0;
      int pad = 0;
      while (angl0 < 360 && i < ind)
      {
            double pi = (data[i][0]*data[i][1] / total) * 100;
            double angli = pi * 360/100;


            x3 = x0 + r*cos( (angli + angl0) * M_PI / 180.0);
            y3 = y0 + r*sin( (angli + angl0) * M_PI / 180.0);

            x4 = (!i)? x2 : x3_prev;
            y4 = (!i)? 0  : y3_prev;

            int red = rand() % 256;
            int green = rand() % 256;
            int blue = rand() % 256;
            pb->Canvas->Pen->Color = (TColor)RGB(red, green, blue);
            TColor color = (TColor)RGB(red, green, blue);
            pb->Canvas->Brush->Color = color;
            pb->Canvas->FillRect(Rect(100, 120+pad, 110 , 130 +
pad));
            pb->Canvas->Pie(x1, y1, x2, y2, x3, y3, x4, y4);

            pb->Canvas->Brush->Color = clWhite;
            int labelX = 120;
            int labelY = 120+pad;
```

```
            pb->Canvas->Font->Size = 10;
            pb->Canvas->Font->Name = "Arial";
            char labelT[LEN];
            sprintf(labelT, "%d, %.0lf%%", data[i][0], pi);
            pb->Canvas->TextOut(labelX, labelY, labelT);

            x3_prev = x3;
            y3_prev = y3;

            angl0 += angli;
            pad += 40;
            i++;
        }

        pb->Canvas->Font->Size = 14; // Размер шрифта
        pb->Canvas->Font->Name = "Arial"; // Название шрифта
        pb->Canvas->Brush->Color = clWhite; // Цвет фона (если нужно)
        pb->Canvas->Pen->Color = clBlack; // Цвет текста

        pb->Canvas->TextOut(x0 - 200, y0-270, "Диаграмма годов
выпуска автомобилей, %");

}

void __fastcall TLinkMainForm::ClearPbClick(TObject *Sender)
{
        pb->Canvas->Brush->Color = clWhite;
        pb->Canvas->FillRect(pb->ClientRect);
}
```

Файл CarList.txt

```
FUEL Toyota_Camry 2023 3000000 Gasoline 60 8.0
FUEL Honda_Civic 2022 2500000 Gasoline 50 7.0
FUEL Ford_Focus 2021 2000000 Gasoline 55 7.5
FUEL Volkswagen_Golf 2023 2800000 Diesel 55 5.5
FUEL BMW_3 2024 4500000 Gasoline 65 8.5
FUEL Mercedes-Benz_C 2024 5000000 Gasoline 70 9.0
FUEL Audi_A4 2023 4000000 Diesel 60 6.0
FUEL Nissan_Altima 2022 2200000 Gasoline 58 7.8
FUEL Hyundai_Sonata 2023 2600000 Gasoline 62 8.2
FUEL Kia_K5 2022 2400000 Gasoline 56 7.6
ELECT Tesla_Model3 2023 5000000 Electric 75 8
ELECT Tesla_ModelS 2024 10000000 Electric 100 10
ELECT BMW_i4 2023 6000000 Electric 80 7
ELECT Audi_etron 2022 7000000 Electric 95 9
ELECT Hyundai_Ioniq_5 2023 4500000 Electric 77 6
ELECT KiaEV6 2022 4200000 Electric 77 7
ELECT Ford_Mustang_E 2023 5500000 Electric 88 8
ELECT Nissan_Ariya 2023 4800000 Electric 87 8
ELECT Volkswagen_ID4 2023 4000000 Electric 77 7
```