МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ   
РОССИЙСКОЙ ФЕДЕРАЦИИ

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Российский государственный университет им. А.Н. Косыгина   
(Технологии. Дизайн. Искусство)»**

Институт ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ И ЦИФРОВЫЕ ТРАНСФОРМАЦИИ

**Отчет по лабораторной работе № 11**

**по дисциплине «Объектно-ориентированное программирование»**

**Тема: «Классы. Наследование и виртуальные методы»**

Выполнил: Сидоров Д. С., группа ИТС-123

Проверил: к.т.н., доц. Семёнов А. А.

Москва, 2025г

**Классы. Наследование и виртуальные методы**

Лабораторная работа посвящена изучению классов в ООП на C++. Перед выполнением лабораторной работы рекомендуется внимательно ознакомиться с лекцией 8. Особое внимание уделите рассмотренному в ней примеру класса Tovar, в котором разбираются указатели и виртуальные методы.

Задание:

Изучить и повторить проиллюстрированный ниже пример, демонстрирующий применение технологии наследования с виртуальными методами.

Взяв за основу проект из лабораторной работы 10, мы создадим его улучшенную версию, которая позволит работать не с двумя массивами, а с одним универсальным, который может хранить данные как продуктовых товаров, так и промышленных. Также реализуем сохранение данных в текстовый файл и считывание их из него в компонент StringGrid (по аналогии с предыдущей л/р). Также реализуем нормальное функционирование кнопки "Общая стоимость", которая см ожет рассчитывать общую стоимость как добавленных вручную товаров, так и загруженных из файла.

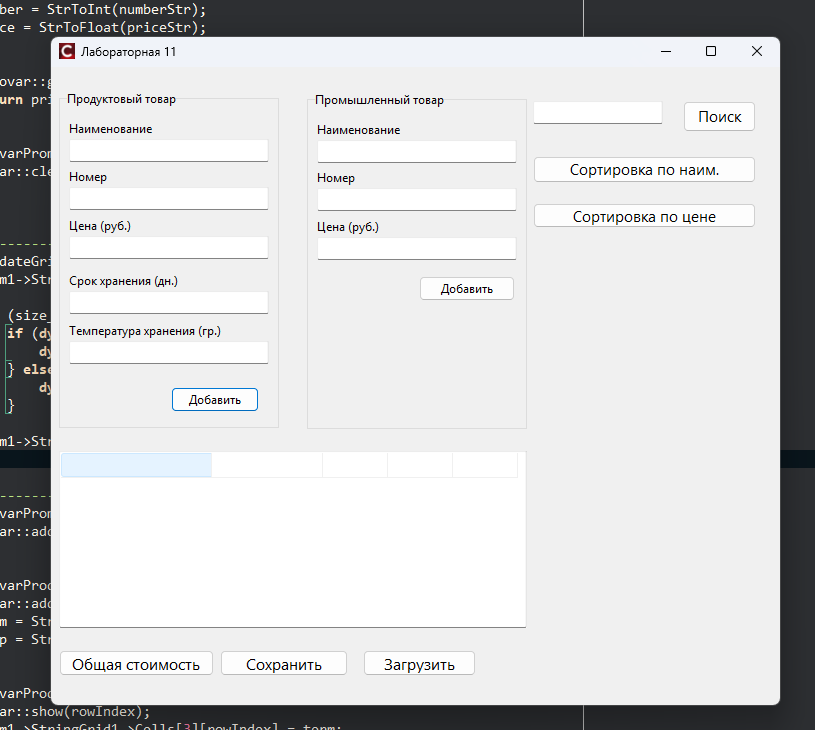


Рисунок 1. Работающий вариант улучшенной программы

**Листинг кода**

**Unit1.cpp**

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

#include <vcl.h>

#include <fstream>

#include <vector>

#include <algorithm>

#include <System.StrUtils.hpp>

#pragma hdrstop

#include "Unit1.h"

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

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TForm1 \*Form1;

std::vector<Tovar\*> allTovars;

const int MAX = 200;

void Tovar::show(int rowIndex) {

Form1->StringGrid1->Cells[0][rowIndex] = name;

Form1->StringGrid1->Cells[1][rowIndex] = number;

Form1->StringGrid1->Cells[2][rowIndex] = price;

}

void Tovar::clear(int flag) {

if(flag == 0) {

Form1->LabeledEdit1->Clear();

Form1->LabeledEdit2->Clear();

Form1->LabeledEdit3->Clear();

}

if(flag == 1) {

Form1->LabeledEdit6->Clear();

Form1->LabeledEdit7->Clear();

Form1->LabeledEdit8->Clear();

}

}

void Tovar::add\_rec(int flag) {

if(flag == 0) {

name = Form1->LabeledEdit1->Text;

number = StrToInt(Form1->LabeledEdit2->Text);

price = StrToFloat(Form1->LabeledEdit3->Text);

}

if(flag == 1) {

name = Form1->LabeledEdit6->Text;

number = StrToInt(Form1->LabeledEdit7->Text);

price = StrToFloat(Form1->LabeledEdit8->Text);

}

}

void Tovar::add\_grid(const String& nameStr, const String& numberStr, const String& priceStr) {

name = nameStr;

number = StrToInt(numberStr);

price = StrToFloat(priceStr);

}

float Tovar::get\_price() {

return price;

}

void TovarProm::clear() {

Tovar::clear(1);

}

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

void UpdateGrid(){

Form1->StringGrid1->RowCount = allTovars.size();

for (size\_t i = 0; i < allTovars.size(); i++) {

if (dynamic\_cast<TovarProd\*>(allTovars[i])) {

dynamic\_cast<TovarProd\*>(allTovars[i])->show(i);

} else {

dynamic\_cast<TovarProm\*>(allTovars[i])->show(i);

}

}

Form1->StringGrid1->Repaint();

}

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

void TovarProm::add\_rec() {

Tovar::add\_rec(1);

}

void TovarProd::add\_rec() {

Tovar::add\_rec(0);

term = StrToInt(Form1->LabeledEdit4->Text);

temp = StrToInt(Form1->LabeledEdit5->Text);

}

void TovarProd::show(int rowIndex) {

Tovar::show(rowIndex);

Form1->StringGrid1->Cells[3][rowIndex] = term;

Form1->StringGrid1->Cells[4][rowIndex] = temp;

}

void TovarProd::add\_grid(const String& nameStr, const String& numberStr, const String& priceStr, const String& termStr, const String& tempStr) {

Tovar::add\_grid(nameStr, numberStr, priceStr);

term = StrToInt(termStr);

temp = StrToInt(tempStr);

}

void TovarProd::clear() {

Tovar::clear(0);

Form1->LabeledEdit4->Clear();

Form1->LabeledEdit5->Clear();

}

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

\_\_fastcall TForm1::TForm1(TComponent\* Owner)

: TForm(Owner)

{

}

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

void \_\_fastcall TForm1::Button1Click(TObject \*Sender)

{

if (allTovars.size() >= MAX) {

ShowMessage(L"Достигнуто максимальное количество товаров!");

return;

}

TovarProd\* prod = new TovarProd;

prod->add\_rec();

allTovars.push\_back(prod);

UpdateGrid();

prod->clear();

}

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

void \_\_fastcall TForm1::Button2Click(TObject \*Sender)

{

if (allTovars.size() >= MAX) {

ShowMessage(L"Достигнуто максимальное количество товаров!");

return;

}

TovarProm\* prom = new TovarProm;

prom->add\_rec();

allTovars.push\_back(prom);

UpdateGrid();

prom->clear();

}

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

void \_\_fastcall TForm1::Button3Click(TObject \*Sender)

{

if(allTovars.empty()) {

ShowMessage(L"Нет товаров!");

} else {

float sum = 0.0;

for (unsigned int e = 0; e < allTovars.size(); e++) {

sum += allTovars[e]->get\_price();

}

ShowMessage(L"Общая стоимость: " + FloatToStr(sum));

}

}

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

void \_\_fastcall TForm1::Button4Click(TObject \*Sender)

{

if(SaveDialog1->Execute())

{

String filename = SaveDialog1->FileName;

TStringList \*list = new TStringList();

for (int i = 0; i < StringGrid1->RowCount; i++)

{

StringGrid1->Rows[i]->Delimiter = L' ';

String line = StringGrid1->Rows[i]->DelimitedText;

list->Add(line);

}

list->SaveToFile(filename, TEncoding::UTF8);

delete list;

}

}

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

void \_\_fastcall TForm1::FormCreate(TObject \*Sender)

{

StringGrid1->FixedCols = 0;

StringGrid1->FixedRows = 0;

StringGrid1->ColCount = 5;

StringGrid1->RowCount = 0;

StringGrid1->ColWidths[0] = 150;

StringGrid1->ColWidths[1] = 110;

StringGrid1->Options << goRowSelect << goAlwaysShowEditor;

SaveDialog1->Filter = L"Текстовые файлы (\*.txt)|\*.txt|Все файлы (\*.\*)|\*.\*";

OpenDialog1->Filter = L"Текстовые файлы (\*.txt)|\*.txt|Все файлы (\*.\*)|\*.\*";

}

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

void \_\_fastcall TForm1::Button5Click(TObject \*Sender)

{

if(OpenDialog1->Execute())

{

String filename = OpenDialog1->FileName;

TStringList \*list = new TStringList();

try

{

try {

list->LoadFromFile(filename, TEncoding::UTF8);

}

catch (...) {

list->LoadFromFile(filename, TEncoding::GetEncoding(1251));

}

StringGrid1->RowCount = (StringGrid1->FixedRows > 0) ? 1 : 0;

int i = 0;

for(int k = 0; k < list->Count; k++)

{

String line = list->Strings[k].Trim();

if (line.IsEmpty()) continue;

TStringList \*parts = new TStringList();

parts->Delimiter = L' ';

parts->DelimitedText = line;

if(parts->Count == 3 && i < MAX)

{

TovarProm\* prom = new TovarProm;

prom->add\_grid(parts->Strings[0], parts->Strings[1], parts->Strings[2]);

prom->show(i);

i++;

allTovars.push\_back(prom);

StringGrid1->RowCount++;

}

else if(parts->Count == 5 && i < MAX)

{

TovarProd\* prod = new TovarProd;

prod->add\_grid(parts->Strings[0], parts->Strings[1], parts->Strings[2],

parts->Strings[3], parts->Strings[4]);

prod->show(i);

i++;

allTovars.push\_back(prod);

StringGrid1->RowCount++;

}

delete parts;

}

UpdateGrid();

}

\_\_finally

{

delete list;

}

}

}

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

void \_\_fastcall TForm1::FormClose(TObject \*Sender, TCloseAction &Action)

{

for (std::vector<Tovar\*>::iterator it = allTovars.begin(); it != allTovars.end(); ++it)

{

delete \*it;

}

allTovars.clear();

}

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

void \_\_fastcall TForm1::Button6Click(TObject \*Sender)

{

if (StringGrid1->RowCount <= 0)

{

ShowMessage(L"Нет данных для сортировки!");

return;

}

std::vector<TStringList\*> gridData;

for (int i = 0; i < StringGrid1->RowCount; i++)

{

TStringList\* row = new TStringList();

for (int j = 0; j < StringGrid1->ColCount; j++)

{

row->Add(StringGrid1->Cells[j][i]);

}

gridData.push\_back(row);

}

for (size\_t i = 0; i < gridData.size() - 1; i++)

{

for (size\_t j = 0; j < gridData.size() - i - 1; j++)

{

if (gridData[j]->Strings[0] > gridData[j+1]->Strings[0])

{

std::swap(gridData[j], gridData[j+1]);

}

}

}

StringGrid1->RowCount = 0;

StringGrid1->RowCount = gridData.size();

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

{

for (int j = 0; j < gridData[i]->Count && j < StringGrid1->ColCount; j++)

{

StringGrid1->Cells[j][i] = gridData[i]->Strings[j];

}

delete gridData[i];

}

}

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

void \_\_fastcall TForm1::Button7Click(TObject \*Sender)

{

if (StringGrid1->RowCount <= 0)

{

ShowMessage(L"Нет данных для сортировки!");

return;

}

std::vector<TStringList\*> gridData;

for (int i = 0; i < StringGrid1->RowCount; i++)

{

TStringList\* row = new TStringList();

for (int j = 0; j < StringGrid1->ColCount; j++)

{

row->Add(StringGrid1->Cells[j][i]);

}

gridData.push\_back(row);

}

for (size\_t i = 0; i < gridData.size() - 1; i++)

{

for (size\_t j = 0; j < gridData.size() - i - 1; j++)

{

float price1 = StrToFloatDef(gridData[j]->Strings[2], 0.0f);

float price2 = StrToFloatDef(gridData[j+1]->Strings[2], 0.0f);

if (price1 > price2)

{

std::swap(gridData[j], gridData[j+1]);

}

}

}

StringGrid1->RowCount = 0;

StringGrid1->RowCount = gridData.size();

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

{

for (int j = 0; j < gridData[i]->Count && j < StringGrid1->ColCount; j++)

{

StringGrid1->Cells[j][i] = gridData[i]->Strings[j];

}

delete gridData[i];

}

}

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

void \_\_fastcall TForm1::Button8Click(TObject \*Sender)

{

if (StringGrid1->RowCount <= 0)

{

ShowMessage(L"Таблица пуста! Нет данных для поиска.");

return;

}

UnicodeString searchText = Edit1->Text.Trim();

if (searchText.IsEmpty())

{

ShowMessage(L"Введите текст для поиска в поле Edit1!");

return;

}

bool found = false;

UnicodeString results = L"Результаты поиска '" + searchText + L"':\n\n";

// Поиск по всем ячейкам

for (int row = 0; row < StringGrid1->RowCount; row++)

{

bool rowFound = false;

UnicodeString rowData;

for (int col = 0; col < StringGrid1->ColCount; col++)

{

if (StringGrid1->Cells[col][row].Pos(searchText) > 0)

{

if (!rowFound)

{

rowData = L"Строка " + IntToStr(row + 1) + L": ";

rowFound = true;

found = true;

}

rowData += StringGrid1->Cells[col][row];

}

}

if (rowFound)

{

results += rowData + L"\n";

}

}

if (found)

{

ShowMessage(results);

}

else

{

ShowMessage(L"Значение '" + searchText + L"' не найдено в таблице.");

}

}

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

**Unit1.h**

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

#ifndef Unit1H

#define Unit1H

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

#include <System.Classes.hpp>

#include <Vcl.Controls.hpp>

#include <Vcl.StdCtrls.hpp>

#include <Vcl.Forms.hpp>

#include <Vcl.Dialogs.hpp>

#include <Vcl.ExtCtrls.hpp>

#include <Vcl.Grids.hpp>

#include <Vcl.Mask.hpp>

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

class TForm1 : public TForm

{

\_\_published: // IDE-managed Components

TGroupBox \*GroupBox1;

TGroupBox \*GroupBox2;

TLabeledEdit \*LabeledEdit2;

TLabeledEdit \*LabeledEdit3;

TLabeledEdit \*LabeledEdit4;

TStringGrid \*StringGrid1;

TButton \*Button1;

TButton \*Button2;

TButton \*Button3;

TButton \*Button4;

TButton \*Button5;

TOpenDialog \*OpenDialog1;

TSaveDialog \*SaveDialog1;

TLabeledEdit \*LabeledEdit1;

TLabeledEdit \*LabeledEdit5;

TLabeledEdit \*LabeledEdit6;

TLabeledEdit \*LabeledEdit7;

TLabeledEdit \*LabeledEdit8;

void \_\_fastcall FormClose(TObject \*Sender, TCloseAction &Action);

void \_\_fastcall Button7Click(TObject \*Sender);

void \_\_fastcall Button8Click(TObject \*Sender);

TButton \*Button6;

TButton \*Button7;

TButton \*Button8;

TEdit \*Edit1;

void \_\_fastcall Button1Click(TObject \*Sender);

void \_\_fastcall Button2Click(TObject \*Sender);

void \_\_fastcall Button3Click(TObject \*Sender);

void \_\_fastcall FormCreate(TObject \*Sender);

void \_\_fastcall Button4Click(TObject \*Sender);

void \_\_fastcall Button5Click(TObject \*Sender);

void \_\_fastcall Button6Click(TObject \*Sender);

private: // User declarations

public: // User declarations

\_\_fastcall TForm1(TComponent\* Owner);

};

class Tovar {

private:

String name;

int number;

float price;

public:

Tovar() : name(""), number(0), price(0) {};

virtual ~Tovar() {};

virtual void add\_rec(int flag);

virtual void show(int rowIndex);

virtual void clear(int flag);

virtual void add\_grid(const String& nameStr, const String& numberStr, const String& priceStr);

float get\_price();

};

class TovarProm : public Tovar {

public:

void add\_rec();

void clear();

};

class TovarProd : public Tovar {

private:

int term;

int temp;

public:

void add\_rec();

void show(int rowIndex);

void clear();

void add\_grid(const String& nameStr, const String& numberStr, const String& priceStr, const String& termStr, const String& tempStr);

};

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

extern PACKAGE TForm1 \*Form1;

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

#endif

**Вывод:** В результате выполнения данной лабораторной работы был создан усовершенствованный проект на основе предыдущей работы, демонстрирующий эффективное применение виртуальных методов в контексте наследования классов. Ключевым улучшением стала реализация универсального массива указателей на базовый класс, способного хранить объекты различных производных классов, что существенно оптимизировало архитектуру приложения и продемонстрировало принцип полиморфизма в ООП. Доработанная функциональность подсчета общей стоимости товаров, работающая корректно как с добавленными вручную товарами, так и с загруженными из файла, наглядно иллюстрирует преимущества использования виртуальных методов. Реализация сохранения данных в файл и их загрузки с учетом полиморфной природы объектов подчеркивает гибкость разработанной системы и ее способность к расширению. Данная лабораторная работа позволила глубже понять механизмы виртуальных функций, полиморфизма и динамического связывания в C++, что является важным аспектом профессионального программирования на языках с поддержкой ООП.