МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

# «ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ им. Ф. СКОРИНЫ»

Курсовой проект

### «Разработка статистическая отчетности в системе АСУ ГГУ»

ГОМЕЛЬ 2003

# Введение

Статистическая отчетность создана для введения бухгалтерской деятельности (расчета прибыли предприятия, зарплаты персонала, введения бухгалтерского учета, вычисления различных налогов и платежей, составление оплаты за ресурсы и т.д.). При этом для облегчения выполнения данных операций обычно применяют метод разделения всех расчетов на более мелкие. (Так для расчета зарплаты используются статьи расчета налогов, начисления самой платы по нескольким статьям, в зависимости от вида выполненных работ, занимаемой должности, стажа и других составляющих. Кроме этого делаются перечисления в различные фонды, такие как пенсионный. Отнимаются и различные штрафы за нарушения норм и т.д.)

Вот таким сложным образом начисляется заработная плата. Если же мы распределим все операции по отдельным статьям, то в дальнейшем нам не придется заново рассчитывать многие операции. Просто нужно просуммировать те статьи, которые относятся к доходам, и отнять те, которые к расходам. То есть мы получаем как бы справочники по статьям дохода и расхода. В этом и есть удобство системы для создания статистической отчетности. Пользователь выбирает из списка нужные статьи и записывает формулу расчета, а программа вычисляет по данным формулы и значениям таблиц и показывает результат.

# 1 Статистическая отчетность

#####

Назначение **Статистической отчетности** в системе АСУ ГГУ – автоматизированное ведение расчетных экономических операций с распечаткой отчетности. Задачи подсистемы:

##### обеспечение санкционированного доступа к данным различных видов отчетности, их распределение и упорядочивание;

##### просмотр статей расходов и доходов;

##### просмотр данных по дебитам и кредитам;

##### работа с платежными документам: требованиями и поручениями (ввод, создание, поиск, корректировка, печать) на основе статистических данных;

##### отображение текущего состояния счетов (используя расчетные данные статей доходов и расходов);

##### формирование, корректировка и печать различных ведомостей, документов и ордеров;

##### создание статотчетности по различным экономическим операциям;

##### построение различных ордеров и создание квитанций на основе статотчетности;

##### обеспечение целостности информации, возможность резервного копирования и восстановления данных.

Данная статистическая отчетность реализует экономическое понятие о отчетности. Она создана для более быстрой и качественной работы с различного вида экономических операций. При справочниках о статьях доходов и расходов (дебитов, кредитов) пользователь может выбрать те из них, по которым он собирается вести расчеты (осуществлять нужную ему экономическую операцию, как например расчет заработной платы). Система АСУ ГГУ проста в работе и предназначена на различных пользователей. То есть человек может успешно работать с системой вне зависимости от его знаний компьютера. Поэтому любой пользователь может без труда сделать экономическую отчетность о проделанной им работе при помощи данной системы.

#

# 2. Структура базы данных статистической отчетности

##

## 2.1 Состав таблиц БД

В процессе функционирования подсистемы «Банк» используется база данных Statotchet, которая подсоединяется к основной базе данных Bank.

CONNECT «D:\GDB\BANKDB.gdb» user «A» password «a»;

/\*Подсоединяется к основной базе данных. Нужно только указать правильный путь доступа к базе данных Bank.\*/

/\*Создается таблица StatRashod, которая описывает статьи расходов.\*/

CREATE TABLE StatRashod (

Id\_StRd INTEGER NOT NULL,

CodeStRd char(8) NOT NULL UNIQUE,

StatName VARCHAR(30) NOT NULL,

PRIMARY KEY (Id\_StRd));

/\*Создается таблица StatProvodki, которая описывает коды дебитов и кредитов в соответствии с кодом статьи расходв\*/

CREATE TABLE StatProvodki (

Id\_Provodki INTEGER NOT NULL,

CodeProvodki CHAR(8) NOT NULL,

IdDsf INTEGER NOT NULL,

IdCsf INTEGER NOT NULL,

PRIMARY KEY (Id\_Provodki),

FOREIGN KEY (CodeProvodki) REFERENCES StatRashod(CodeStRd));

/\*Создается таблица StatRdDd, которая описывает статьи расходов и доходов.\*/

CREATE TABLE StatRdDd (

Id\_StatRdDd INTEGER NOT NULL,

StatRdDdName VARCHAR(30) NOT NULL,

PRIMARY KEY (Id\_StatRdDd));

/\*Создается индексы и генераторы для таблиц\*/

CREATE UNIQUE INDEX StatRashod\_INDEX ON StatRashod (Id\_StRd);

CREATE UNIQUE INDEX StatRdDd\_INDEX ON StatRdDd (Id\_StatRdDd);

CREATE UNIQUE INDEX StatProvodki\_INDEX ON StatProvodki (Id\_Provodki);

CREATE GENERATOR Id\_StRd\_GEN;

CREATE GENERATOR Id\_StatRdDd\_GEN;

CREATE GENERATOR Id\_Provodki\_GEN;

COMMIT;

set term #;

/\* триггеры для заполнения полей XXX\_ID соответствующих таблиц (все срабатывают до добавления данных в таблицу) \*/

create trigger TRG\_Id\_StRd for StatRashod active before insert position 0 as

begin

new. Id\_StRd = gen\_id (Id\_StRd \_GEN, 1); /\* новое значение XXX\_ID РАВНО старое плюс 1 \*/

end#

create trigger TRG\_Id\_StatRdDd for StatRdDd active before insert position 0 as

begin

new. Id\_StatRdDd = gen\_id (Id\_StatRdDd \_GEN, 1); /\* новое значение XXX\_ID РАВНО старое плюс 1 \*/

end#

create trigger TRG\_Id\_Provodki for Stat Provodki active before insert position 0 as

begin

new. Id\_ Provodki = gen\_id (Id\_ Provodki \_GEN, 1); /\* новое значение XXX\_ID РАВНО старое плюс 1 \*/

end#

set term;#

COMMIT;

## 2.2 Структура данных программы

Таблица StatRashod:

№Field name Type Width Dec

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | Id\_StRd | INTEGER |  | Ключ, Счетчик |
| 2 | CodeStRd | CHAR | 8 | Код Статьи расходов |
| 3 | StatName | VARCHAR | 30 | Название статьи расходов |

Таблица StatProvodki:

№Field name Type Width Dec

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | Id\_Provodki | INTEGER |  | Ключ, Счетчик |
| 2 | CodeProvodki | CHAR | 8 | Код проводки в соответсвии с кодами статей расходов |
| 3 | IdDsf | INTEGER |  | Код дебита |
| 4 | IdСsf | INTEGER |  | Код кредита |

Таблица StatRdDd:

№Field name Type Width Dec

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | Id\_StRdDd | INTEGER |  | Ключ, Счетчик |
| 2 | StatRdDdName | VARCHAR | 30 | Название статьи расходов |

# 3. Схема диалога пользователя с системой АСУ ГГУ и создание статистической отчетности

## 3.1 Схема диалога пользователя с системой по стандарту CUA

Взаимодействие пользователя с подсистемой осуществляется с использованием модели пользовательского интерфейса, соответствующей известному стандарту CUA (Common User Access). Хотя этот стандарт первоначально был предложен фирмой IBM в качестве внутрифирменного стандарта, на сегодняшний день он превратился в общепризнанный стандарт проектирования пользовательского интерфейса. Далее рассматриваются некоторые базовые концепции стандарта CUA.

#### Метафора рабочего стола. В соответствии с этой идеей экран компьютера рассматривается как рабочая поверхность, на которой пользователь и программа размещают визуальную информацию. Информация, относящаяся к определенному виду деятельности, выполняемой в текущий момент времени, организуется в виде окна. Пользователь имеет возможность свободно перемещать окна, изменять их размеры, выбирать текущее окно, перемещая его на передний план, и, наконец, создавать новые и уничтожать ненужные ему окна. Обязательным требованием к интерфейсу является многооконность-то есть возможность создания пользователем в рабочей области нескольких окон.

Меню-ориентированный интерфейс. CUA жестко регламентирует требования к организации пользовательского меню. Это размещение главного меню в верхней строке экрана, организация «выпадающих» подменю, возможность выбора требуемой позиции с помощью клавиатуры и манипулятора типа «мышь», наличие «горячих» клавиш для позиций меню. Средства организации меню должны обеспечивать возможность различной визуализации активных и неактивных позиций, а также организовывать связь с системой помощи, которая позволяет получить подсказку для каждой позиции меню.

Диалоговые окна. В соответствии с концепцией CUA общение пользователя с программой (ввод данных пользователем или вывод сообщений программой) осуществляется с помощью так называемых «всплывающих» (pop-up) диалоговых окон. Такие окна содержат маркированные области, в которые пользователь может осуществлять ввод данных, выбор опций программы, просмотр информации и другие операции. Важным элементом диалоговых окон являются так называемые органы управления – кнопки, полосы прокрутки, позволяющие пользователю управлять поведением программы или просмотром информации.

Контекстно-зависимая помощь. Программа, удовлетворяющая требованиям CUA, обязана поддерживать систему помощи, содержание которой зависит от текущей выполняемой пользователем операции. Это касается не только вывода справочной информации по запросу пользователя, но также и наличия кратких подсказок по используемым клавишам, идентификации текущей выполняемой функции или операции.

## 3.2 Схема диалога пользователя с системой АСУ ГГУ

Реализация системы АСУ ГГУ проведена на языке C++Bilder5 с применением объектно-ориентированной технологии, обеспечивающей разработку пользовательского интерфейса для прикладных программ в соответствии с изложенными выше концепциями стандарта CUA. Взаимодействие пользователя с подсистемой осуществляется в основном с использованием диалоговых окон, содержащих следующие основные органы управления: «Файл», «Проводки», «НДС», «Банк», «Касса», «Питание», «Услуги», «Материалы», «Командировки», «Квитанции», «Документы», «Справочники». Некоторые из них содержат другие диалоговые окна. Например: диалоговое окно «Файл» кроме пункта «Создание данных за месяц» содержит так же подменю «Настройка», «Служебные функции», «Параметры работы».

## 3.3 Реализация статистической отчетности в системе АСУ ГГУ

Для реализации статистической отчетности были созданы три формы написанные на языке C++Bilder5. Первая описывает статьи расходов. Вторая соответствие проводков со статьями расходов. Наконец третья описывает статьи расходов / доходов. Эта третья является родительской.

Для создания этих форм были созданы компоненты AsuTable: StatRashodTable, StatRdDdTable, StatProvodkiTable. Эти компоненты реализованы программами StatRashodTable.cpp, StatRashodTable.h, StatProvodkiTable.cpp, StatProvodkiTable.h, StatProvodkiTable.cpp, StatProvodkiTable.h. Текст программ приведен ниже в Приложении А, В и С.

Кроме того в создании использовались кнопки «найти» и «выход». Данные кнопки реализованы как методы Searchform из программы SearchWn.h. При нажатии манипулятором мышь на кнопку, появляется новая форма для нахождения нужной вам информации. При задействовании кнопки «выход» данная форма закрывается.

Данные в форму заносятся из таблиц. Обновляются и изменяются при помощи метода AsuDbgrid.

Программы реализующие эти формы: USStatRashodForm.h, USStatRashodForm.cpp, USStatRashodForm.dfm, USStatRdDdForm.h, USStatRdDdForm.cpp, USStatRdDdForm.dfm, USStatProvodkiForm.h, USStatProvodkiForm.cpp, USStatProvodkiForm.dfm. Тексты программ приведены в приложениях А, В и С.

# Заключение

Статистическая отчетность создана для введения бухгалтерской деятельности (расчета прибыли предприятия, зарплаты персонала, введения бухгалтерского учета, вычисления различных налогов и платежей, составление оплаты за ресурсы и т.д.). Данная статистическая отчетность в системе АСУ ГГУ реализует экономическое понятие о отчетности. Она создана для более быстрой и качественной работы с различного вида экономических операций. При справочниках о статьях доходов и расходов (дебитов, кредитов) пользователь может выбрать те из них, по которым он собирается вести расчеты (осуществлять нужную ему экономическую операцию, как например расчет заработной платы). Система АСУ ГГУ проста в работе и предназначена на различных пользователей. То есть человек может успешно работать с системой вне зависимости от его знаний компьютера. Поэтому любой пользователь может без труда сделать экономическую отчетность о проделанной им работе при помощи данной системы.

**Приложение А**

**Программа USStatRashodForm.h**

// –

#ifndef USStatRashodFormH

#define USStatRashodFormH

 // –

#include <Classes.hpp>

#include <Controls.hpp>

#include <StdCtrls.hpp>

#include <Forms.hpp>

#include «ASUBUTEX.h»

#include «AsuDBGrd.h»

#include «AsuTable.h»

#include «AsuForm.h»

#include «StatRashodTable.h»

#include <Db.hpp>

#include <DBGrids.hpp>

#include <DBTables.hpp>

#include <Grids.hpp>

#include «ASUQUERY.h»

 // –

class TFSStatRashod: public AsuForm

{

\_\_published:  // IDE-managed Components

TButton \*SearchButton;

AsuButtonExit \*AsuButtonExit1;

StatRashodTable \*StatRashodTable1;

AsuQuery \*AsuQuery1;

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

void \_\_fastcall AsuButtonExit1Click (TObject \*Sender);

private:  // User declarations

public: // User declarations

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

};

 // –

extern PACKAGE TFSStatRashod \*FSStatRashod;

 // –

#endif

**Программа USStatRashodForm.cpp**

 // –

#include <vcl.h>

#pragma hdrstop

#include «AsuObj.h»

#include «USStatRashodForm.h»

 // –

#pragma package (smart\_init)

#pragma link «ASUBUTEX»

#pragma link «AsuDBGrd»

#pragma link «AsuTable»

#pragma link «StatRashodTable»

#include «SearchWn.h»

#pragma link «ASUQUERY»

#pragma resource «\*.dfm»

TFSStatRashod \*FSStatRashod;

 // –

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

: AsuForm(Owner)

{

}

 // –

void \_\_fastcall TFSStatRashod: SearchButtonClick (TObject \*Sender)

{

SearchForm->searchList = NULL;

SearchForm->gridSearch = StatRashodDBGrid;

SearchForm->ShowModal();

}

 // –

void \_\_fastcall TFSStatRashod: AsuButtonExit1Click (TObject \*Sender)

{

((AsuForm \*) Owner)->Close();

}

 // –

**Программа USStatRashodForm.dfm**

object FSStatRashod: TFSStatRashod

Left = 194

Top = 9

Width = 544

Height = 501

Caption = 'Статья расходов'

Color = clBtnFace

Font. Charset = DEFAULT\_CHARSET

Font. Color = clWindowText

Font. Height = -11

Font. Name = 'MS Sans Serif'

Font. Style = []

OldCreateOrder = False

PixelsPerInch = 96

TextHeight = 13

object StatRashodDBGrid: AsuDBGrid

Tag = 4

Left = 0

Top = 0

Width = 536

Height = 417

Align = alTop

TabOrder = 0

TitleFont. Charset = DEFAULT\_CHARSET

TitleFont. Color = clWindowText

TitleFont. Height = -11

TitleFont. Name = 'MS Sans Serif'

TitleFont. Style = []

AsuDataSet = StatRashodTable1

Update = True

end

object SearchButton: TButton

Left = 16

Top = 432

Width = 75

Height = 25

Caption = '&Найти'

TabOrder = 1

OnClick = SearchButtonClick

end

object AsuButtonExit1: AsuButtonExit

Left = 144

Top = 432

Width = 75

Height = 25

Cancel = True

Caption = 'Выход'

TabOrder = 2

OnClick = AsuButtonExit1Click

end

object StatRashodTable1: StatRashodTable

Exclusive = True

TableName = 'StatRashod'

Left = 232

Top = 72

end

object AsuQuery1: AsuQuery

SQL. Strings = (

'Select CodeStRd, StatName from'

' StatRashod'

'Order by CodeStRd')

Left = 184

Top = 104

end

end

**Программа StatRashodTable.h**

// –

#ifndef StatRashodTableH

#define StatRashodTableH

 // –

#include <SysUtils.hpp>

#include <Controls.hpp>

#include <Classes.hpp>

#include <Forms.hpp>

#include «AsuTable.h»

#include <Db.hpp>

#include <DBTables.hpp>

 // –

class PACKAGE StatRashodTable: public AsuTable

{

private:

char \*getNameColumn (AnsiString nameField);

TFieldType getFieldType(AnsiString);

protected:

public:

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

\_\_published:

};

 // –

#endif

**Программа StatRashodTable.h**

// –

#include <vcl.h>

#pragma hdrstop

#include «AsuObj.h»

#include «StatRashodTable.h»

 // #include «OValKursTp.h»

#pragma link «AsuTable»

#pragma package (smart\_init)

 // –

 // ValidCtrCheck is used to assure that the components created do not have

 // any pure virtual functions.

static inline void ValidCtrCheck (StatRashodTable \*)

{

new StatRashodTable(NULL);

}

 // –

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

: AsuTable(Owner)

{

TableName = «StatRashod»;

}

 // –

char \*StatRashodTable:getNameColumn (AnsiString nameField)

{

if (! nameField. AnsiCompareIC («CodeStRd»)) return «Код Статьи расходов»;

if (! nameField. AnsiCompareIC («StatName»)) return «Наименование статьи»;

return 0;

}

 // –

TFieldType StatRashodTable:getFieldType (AnsiString nameField)

{

return ftString;

}

 // –

namespace Statrashodtable

{

void \_\_fastcall PACKAGE Register()

{

TComponentClass classes[1] = {\_\_classid(StatRashodTable)};

RegisterComponents («АСУ-таблицы», classes, 0);

}

}

#

# Приложение B

**Программа USStatRdDdForm.h**

// –

#ifndef USStatRdDdFormH

#define USStatRdDdFormH

 // –

#include <Classes.hpp>

#include <Controls.hpp>

#include <StdCtrls.hpp>

#include <Forms.hpp>

#include «ASUBUTEX.h»

#include «AsuDBGrd.h»

#include «AsuTable.h»

#include «AsuForm.h»

#include «StatRdDdTable.h»

#include <Db.hpp>

#include <DBGrids.hpp>

#include <DBTables.hpp>

#include <Grids.hpp>

#include «ASUQUERY.h»

 // –

class TFSStatRdDd: public AsuForm

{

\_\_published:  // IDE-managed Components

TButton \*SearchButton;

AsuButtonExit \*AsuButtonExit1;

StatRdDdTable \*StatRdDdTable1;

AsuQuery \*AsuQuery1;

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

void \_\_fastcall AsuButtonExit1Click (TObject \*Sender);

private:  // User declarations

public: // User declarations

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

};

 // –

extern PACKAGE TFSStatRdDd \*FSStatRdDd;

 // –

#endif

**Программа USStatRdDdForm.cpp**

// –

#include <vcl.h>

#pragma hdrstop

#include «AsuObj.h»

#include «USStatRdDdForm.h»

 // –

#pragma package (smart\_init)

#pragma link «ASUBUTEX»

#pragma link «AsuDBGrd»

#pragma link «AsuTable»

#pragma link «StatRdDdTable»

#include «SearchWn.h»

#pragma link «ASUQUERY»

#pragma resource «\*.dfm»

TFSStatRdDd \*FSStatRdDd;

 // –

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

: AsuForm(Owner)

{

}

 // –

void \_\_fastcall TFSStatRdDd: SearchButtonClick (TObject \*Sender)

{

SearchForm->searchList = NULL;

SearchForm->gridSearch = StatRdDdDBGrid;

SearchForm->ShowModal();

}

 // –

void \_\_fastcall TFSStatRdDd: AsuButtonExit1Click (TObject \*Sender)

{

((AsuForm \*) Owner)->Close();

}

 // –

**Программа USStatRdDdForm.dfm**

bject FSStatRdDd: TFSStatRdDd

Left = 194

Top = 9

Width = 544

Height = 501

Caption = 'Статьи расходов\доходов'

Color = clBtnFace

Font. Charset = DEFAULT\_CHARSET

Font. Color = clWindowText

Font. Height = -11

Font. Name = 'MS Sans Serif'

Font. Style = []

OldCreateOrder = False

PixelsPerInch = 96

TextHeight = 13

object StatRdDdDBGrid: AsuDBGrid

Tag = 4

Left = 0

Top = 0

Width = 536

Height = 417

Align = alTop

TabOrder = 0

TitleFont. Charset = DEFAULT\_CHARSET

TitleFont. Color = clWindowText

TitleFont. Height = -11

TitleFont. Name = 'MS Sans Serif'

TitleFont. Style = []

AsuDataSet = StatRdDdTable1

Update = True

end

object SearchButton: TButton

Left = 16

Top = 432

Width = 75

Height = 25

Caption = '&Найти'

TabOrder = 1

OnClick = SearchButtonClick

end

object AsuButtonExit1: AsuButtonExit

Left = 144

Top = 432

Width = 75

Height = 25

Cancel = True

Caption = 'Выход'

TabOrder = 2

OnClick = AsuButtonExit1Click

end

object StatRdDdTable1: StatRdDdTable

Exclusive = True

TableName = 'StatRdDd'

Left = 232

Top = 72

end

object AsuQuery1: AsuQuery

SQL. Strings = (

'Select StatRdDdName from'

' StatRdDd'

'Order by StatRdDdName')

Left = 184

Top = 104

end

end

**Программа StatRdDdTable.h**

 // –

#ifndef StatRdDdTableH

#define StatRdDdTableH

 // –

#include <SysUtils.hpp>

#include <Controls.hpp>

#include <Classes.hpp>

#include <Forms.hpp>

#include «AsuTable.h»

#include <Db.hpp>

#include <DBTables.hpp>

 // –

class PACKAGE StatRdDdTable: public AsuTable

{

private:

char \*getNameColumn (AnsiString nameField);

TFieldType getFieldType(AnsiString);

protected:

public:

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

\_\_published:

};

 // –

#endif

**Программа StatRdDdTable.cpp**

 // –

#include <vcl.h>

#pragma hdrstop

#include «AsuObj.h»

#include «StatRdDdTable.h»

#pragma link «AsuTable»

#pragma package (smart\_init)

 // –

 // ValidCtrCheck is used to assure that the components created do not have

 // any pure virtual functions.

static inline void ValidCtrCheck (StatRdDdTable \*)

{

new StatRdDdTable(NULL);

}

 // –

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

: AsuTable(Owner)

{

TableName = «StatRashod/Dohod»;

}

 // –

char \*StatRdDdTable:getNameColumn (AnsiString nameField)

{

if (! nameField. AnsiCompareIC («StatRdDdName»)) return «наименование статьи»;

return 0;

}

 // –

TFieldType StatRdDdTable:getFieldType (AnsiString nameField)

{

return ftString;

}

 // –

namespace StatRdDdtable

{

void \_\_fastcall PACKAGE Register()

{

TComponentClass classes[1] = {\_\_classid(StatRdDdTable)};

RegisterComponents («АСУ-таблицы», classes, 0);

}

}

 // –

#

# Приложение C

**Программа USStatProvodkiForm.h**

// –

#ifndef USStatProvodkiFormH

#define USStatProvodkiFormH

 // –

#include <Classes.hpp>

#include <Controls.hpp>

#include <StdCtrls.hpp>

#include <Forms.hpp>

#include «ASUBUTEX.h»

#include «AsuDBGrd.h»

#include «AsuTable.h»

#include «AsuForm.h»

#include «StatProvodkiTable.h»

#include <Db.hpp>

#include <DBGrids.hpp>

#include <DBTables.hpp>

#include <Grids.hpp>

#include «ASUQUERY.h»

 // –

class TFSStatProvodki: public AsuForm

{

\_\_published:  // IDE-managed Components

TButton \*SearchButton;

AsuButtonExit \*AsuButtonExit1;

StatProvodkiTable \*StatProvodkiTable1;

AsuQuery \*AsuQuery1;

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

void \_\_fastcall AsuButtonExit1Click (TObject \*Sender);

private:  // User declarations

public: // User declarations

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

};

 // –

extern PACKAGE TFSStatProvodki \*FSStatProvodki;

 // –

#endif

**Программа USStatProvodkiForm.cpp**

// –

#include <vcl.h>

#pragma hdrstop

#include «AsuObj.h»

#include «USStatProvodkiForm.h»

 // –

#pragma package (smart\_init)

#pragma link «ASUBUTEX»

#pragma link «AsuDBGrd»

#pragma link «AsuTable»

#pragma link «StatProvodkiTable»

#include «SearchWn.h»

#pragma link «ASUQUERY»

#pragma resource «\*.dfm»

TFSStatRashod \*FSStatProvodki;

 // –

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

: AsuForm(Owner)

{

}

 // –

void \_\_fastcall TFSStatProvodki: SearchButtonClick (TObject \*Sender)

{

SearchForm->searchList = NULL;

SearchForm->gridSearch = StatProvodkiDBGrid;

SearchForm->ShowModal();

}

 // –

void \_\_fastcall TFSStatProvodki: AsuButtonExit1Click (TObject \*Sender)

{

((AsuForm \*) Owner)->Close();

}

 // –

**Программа USStatProvodkiForm.dfm**

object FSStatProvodki: TFSStatProvodki

Left = 194

Top = 9

Width = 544

Height = 501

Caption = 'Проводки'

Color = clBtnFace

Font. Charset = DEFAULT\_CHARSET

Font. Color = clWindowText

Font. Height = -11

Font. Name = 'MS Sans Serif'

Font. Style = []

OldCreateOrder = False

PixelsPerInch = 96

TextHeight = 13

object StatProvodkiDBGrid: AsuDBGrid

Tag = 4

Left = 0

Top = 0

Width = 536

Height = 417

Align = alTop

TabOrder = 0

TitleFont. Charset = DEFAULT\_CHARSET

TitleFont. Color = clWindowText

TitleFont. Height = -11

TitleFont. Name = 'MS Sans Serif'

TitleFont. Style = []

AsuDataSet = StatProvodkiTable1

Update = True

end

object SearchButton: TButton

Left = 16

Top = 432

Width = 75

Height = 25

Caption = '&Найти'

TabOrder = 1

OnClick = SearchButtonClick

end

object AsuButtonExit1: AsuButtonExit

Left = 144

Top = 432

Width = 75

Height = 25

Cancel = True

Caption = 'Выход'

TabOrder = 2

OnClick = AsuButtonExit1Click

end

object StatProvodkiTable1: StatProvodkiTable

Exclusive = True

TableName = 'StatProvodki'

Left = 232

Top = 72

end

object AsuQuery1: AsuQuery

SQL. Strings = (

'Select CodeProvodki, IdDsf, IdCsf from'

' StatProvodki'

'Order by CodeProvodki')

Left = 184

Top = 104

end

end

**Программа StatProvodki.h**

// –

#ifndef StatProvodkiTableH

#define StatProvodkiTableH

 // –

#include <SysUtils.hpp>

#include <Controls.hpp>

#include <Classes.hpp>

#include <Forms.hpp>

#include «AsuTable.h»

#include <Db.hpp>

#include <DBTables.hpp>

 // –

class PACKAGE StatProvodkiTable: public AsuTable

{

private:

char \*getNameColumn (AnsiString nameField);

TFieldType getFieldType(AnsiString);

protected:

public:

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

\_\_published:

};

 // –

#endif

**Программа StatProvodki.h**

// –

#include <vcl.h>

#pragma hdrstop

#include «AsuObj.h»

#include «StatProvodkiTable.h»

#include «OValKursTp.h»

#pragma link «AsuTable»

#pragma package (smart\_init)

 // –

 // ValidCtrCheck is used to assure that the components created do not have

 // any pure virtual functions.

static inline void ValidCtrCheck (StatProvodkiTable \*)

{

new StatProvodkiTable(NULL);

}

 // –

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

: AsuTable(Owner)

{

TableName = «StatProvodki»;

}

 // –

char \*StatProvodkiTable:getNameColumn (AnsiString nameField)

{

if (! nameField. AnsiCompareIC («CodeProvodki»)) return «Код проводки»;

if (! nameField. AnsiCompareIC («IdDsf»)) return «Номер дебита»;

if (! nameField. AnsiCompareIC («IdСsf»)) return «Номер кредита»;

return 0;

}

 // –

TFieldType StatProvodkiTable:getFieldType (AnsiString nameField)

{

return ftString;

}

namespace StatProvodkiTable

{

void \_\_fastcall PACKAGE Register()

{

TComponentClass classes[1] = {\_\_classid(StatProvodkiTable)};

RegisterComponents («АСУ-таблицы», classes, 0);

}}