**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**«БАРАНОВИЧСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**Факультет** Инженерный

**Кафедра** Информационных систем и технологий

**Дата регистрации работы в деканате \_\_\_\_\_\_\_\_\_**

**Дата регистрации работы на кафедре \_\_\_\_\_\_\_\_\_**

**Отметка о допуске к защите \_\_\_\_\_\_\_\_\_**

**Оценка за защиту \_\_\_\_\_\_\_\_\_**

**КУРСОВАЯ РАБОТА**

**по дисциплине** Основы алгоритмизации и программирования

**Тема: «**Программирование с использованием структур и файлов в среде С++ Builder»

**Исполнитель:**

Инженерный,1курс, ИСТ-11

Галабурда Роман Витольдович

**Руководитель:**

Старший преподаватель кафедры Информационных систем и технологий

Магистр педагогических наук

Раковцы Галина Михайловна

**Барановичи 2015**

**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**«БАРАНОВИЧСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

РЕЦЕНЗИЯ

на курсовую работу

(регистрационный №\_\_\_\_\_)

Студента Галабурды Романа Витольдович

Факультет ИНЖЕНЕРНЫЙ

Курс 1

Дисциплина Основы алгоритмизации и программирования

Рецензент Раковцы Галина Михайловна

Дата получения к/р для рецензирования \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата возвращения к/р после рецензирования \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оценка\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Подпись преподавателя-рецензента \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Текст рецензии:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**РЕФЕРАТ**

Курсовая работа: 49 с., 16 рис., 5 источников, 4 прил., 5 табл.

НАСТОЛЬНОЕ БИЗНЕС-ПРИЛОЖЕНИЕ, СРЕДА BORLAND С++ BUILDER, БАЗА ДАННЫХ СОТРУДНИКОВ ПРЕДПРИЯТИЯ И СОБРАННЫХ ИЗДЕЛИЙ, РАЗРАБОТКА ПРОГРАММЫ.

Предмет исследования - применение среды программирования Borland С++Builder для создания настольного бизнес-приложения, содержащего и обрабатывающего базу данных предприятия.

Объект исследования - среда Borland С++Builder.

Цель работы – написать программу, реализующую создание и обработку базы данных предприятия и последующую печать ведомостей.

Основой для выполнения работы стала справочная, научная и учебная литература.

При выполнении работы использовался метод: анализ научно-методической литературы, проектов.

Областью возможного практического применения данной работы является любое предприятие, производящее продукцию.

Автор подтверждает, что приведенный в работе расчетно-аналитический материал правильно и объективно отражает состояние исследуемого процесса, а все заимствованные из литературных и других источников теоретические, методологические и методические положения и концепции сопровождаются ссылками на их авторов.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись студента)

**СОДЕРЖАНИЕ**

ВВЕДЕНИЕ……………………………………………………………………..…6

1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ…………………..……………………………….9

1.1 Постановка задачи………………………………………………..……9

1.2 Теоретические аспекты предметной области……………………....10

1.3 Алгоритм решения задачи……………………………………………12

1.4 Блок-схема алгоритма ……………………………………………......15

2 ПРАКТИЧЕСКАЯ ЧАСТЬ………………………………………………...…19

2.1 Описание объектов, свойств, методов……………………………….19

2.2 Формы и основные модули текста программы……………………..22

2.3 Результаты тестирования программного продукта…………………29

2.4 Анализ результатов……………….……...…………………………...36

ЗАКЛЮЧЕНИЕ………………………………………………………….………37

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ ………………………..……...38

ПРИЛОЖЕНИЕ А ………………………….…...………………………………39

ПРИЛОЖЕНИЕ Б ……………………………………………………...………..47

ПРИЛОЖЕНИЕ В ……………………………………………………...………..48

ПРИЛОЖЕНИЕ Г……………………………………………………...………...49

**ВВЕДЕНИЕ**

В современности ни одно предприятие не обходится без внутренней базы данных о сотрудниках, изделиях, бухгалтерском учете и т.д. В большинстве своем предприятия заказывают программное обеспечение для создания и обработки таких баз данных. Так как появляются всё новые и новые предприятия, то соответственно каждое из них нуждается в разработке такого программного обеспечения, которое содержало бы в себе функции создания и редактирования базы данных, печати ведомостей и т.д. Поэтому тема данный курсовой работы всегда будет актуальной, так как имеет очень широкое применения во многих областях промышлености и социальной жизни.

Целью данной курсовой работы является разработка настольного бизнес-приложения, содержащего и обрабатывающего базу данных предприятия.

Задачи курсовой работы: в полной мере овладеть возможностями среды C++ Builder для написания настольных приложений.

Структура курсовой работы:

В первой главе данной курсовой работы формируются основные задачи и алгоритм решения, которые помогут решить поставленную задачу. Во второй главе приводится описание компонентов, используемых при реализации практической части курсовой работы, а также этапы разработки программы. Список использованных источников содержит полный перечень литературы, использованной при разработке программы.

Borland C++ Builder сегодня является наиболее совершенной визуальной средой быстрой разработки на языке C++ для Windows. C++ Builder объединяет в себе комплекс объектных библиотек (STL, VCL, CLX, и др.), компилятор, отладчик, редактор кода и многие другие компоненты. Цикл разработки аналогичен Delphi. Большинство компонентов, разработанных в Delphi, можно использовать и в C++ Builder без модификации, но обратное утверждение не верно.

В ее состав входит около 200 самых разных компонентов, а создание законченной программы требует минимума усилий. Ближайший конкурент Borland C++ Builder — это не система Microsoft Visual C++, которая построена по другой схеме, a Microsoft Visual Basic. Однако эффективность программ, создаваемых с помощью C++ Builder, в десятки раз превосходит быстродействие программ, написанных на MS Visual Basic. Да и по числу свободных доступных компонентов равных среде C++ Builder сегодня не найти.В данной курсовой работе для разработки программы использовалась среда программирования Borland C++ Builder.

Borland C++ Builder, является наиболее совершенной визуальной средой быстрой разработки на С++ для Windows. Вместо отдельного инструментария, оперирующего визуальными элементами управления, в C++ Builder интегрирована так называемая Палитра компонент, разделенная картотечными вкладками на множество групп, объединающих в себе совокуности подобных и равноуровневых компонент.

Система содержит библиотеку из более 100 визуальных компонент, имеющих возможность использоваться повторно бесконечное количество раз. Компоненты перетаскиваются мышью на форму и превращаются в интерфейс вашей программы. Помимо известных элементов управления Windows (кнопки, инструменты прокрутки, поля редактирования, простые и комбинированные списки и т.д.) библиотека содержит новые компоненты поддержки диалогов, обслуживания баз данных и многие другие.

После размещения компонент на форме, Инспектор объектов поможет вам устанавливать их свойства и предписывать событиям коды обработки. Ваш проект будет строиться постепенно, на фоне производимых вами изменений в свойствах, событиях и функциях используемых элементов.  Хорошо продумано разделение и редактирование программного модуля по двум его частям: интерфейсной и собственно кодовой.

Благодаря уникальному сочетанию удобства разработки пользовательских интерфейсов, компонентной архитектуры, однотипности доступа к различным базам данным возможно упрощенное программирование в среде разрабоки C++ Builder.

**1.ТЕОРЕТИЧЕСКАЯ ЧАСТЬ**

* 1. **Постановка задачи**

Написать программу на языке С++, реализующую и обрабатывающую базу данных предприятия по изготовлению деталей. Программа должна работать с файлами (база данных) и иметь Windows-интерфейс.

1. Создать файл, содержащий сведения о количестве изделий категорий А, В, С, собранных рабочим за месяц. Структура записей имеет поля:

* фамилия сборщика
* наименование цеха
* количество изделий по категориям, собранных рабочим за месяц.

Количество записей – произвольное.

2. Cчитая, заданными значения расценок  за выполненную работу по сборке единицы изделия категорий А, В, С соответственно, выдать на печать следующую информацию:

- общее количество изделий категорий А, В, С, собранных рабочим цеха Х;

- ведомость зарплаты рабочих цеха Х;

- средний размер зарплаты работников этого цеха,

- фамилию рабочего завода, собравшего наибольшее количество изделий.

* 1. **Теоретические аспекты предметной области**

Для решения поставленной задачи необходимо использовать возможности C++ Builder для ввода и вывода текстовой информации, а также функции работы с файлами.

C++ Builder представляет собой SDI-приложение, главное окно которого содержит настраиваемую инструментальную панель (слева) и палитру компонентов (справа). Помимо этого, по умолчанию при запуске C++ Builder появляются окно инспектора объектов (слева) и форма нового приложения (справа). Под окном формы приложения находится окно редактора кода.

Формы являются основой приложений C++ Builder. Создание пользовательского интерфейса приложения заключается в добавлении в окно формы элементов объектов C++ Builder, называемых компонентами. Компоненты C++ Builder располагаются на палитре компонентов, выполненной в виде многостраничного блокнота. Важная особенность C++ Builder состоит в том, что он позволяет создавать собственные компоненты и настраивать палитру компонентов, а также создавать различные версии палитры компонентов для разных проектов.

 Компоненты разделяются на видимые (визуальные) и невидимые (невизуальные). Визуальные компоненты появляются во время выполнения точно так же, как и во время проектирования. Примерами являются кнопки и редактируемые поля. Невизуальные компоненты появляются во время проектирования как пиктограммы на форме. Они никогда не видны во время выполнения, но обладают определенной функциональностью (например, обеспечивают доступ к данным, вызывают стандартные диалоги Windows 95 и др.)

Свойства являются атрибутами компонента, определяющими его внешний вид и поведение. Многие свойства компонента в колонке свойств имеют значение, устанавливаемое по умолчанию (например, высота кнопок). Свойства компонента отображаются в окне свойств (Properties). Инспектор объектов отображает опубликованные (published) свойства компонентов. Помимо published-свойств, компоненты могут и чаще всего имеют общие (public), опубликованные свойства, которые доступны только во время выполнения приложения. Инспектор объектов используется для установки свойств во время проектирования. Список свойств располагается на странице свойств инспектора объектов. Можно определить свойства во время проектирования или написать код для видоизменения свойств компонента во время выполнения приложения.

Страница событий (Events) инспектора объектов показывает список событий, распознаваемых компонентом (программирование для операционных систем с графическим пользовательским интерфейсом, в частности, для Windows 95 или Windows NT предполагает описание реакции приложения на те или иные события, а сама операционная система занимается постоянным опросом компьютера с целью выявления наступления какого-либо события). Каждый компонент имеет свой собственный набор обработчиков событий. В C++ Builder следует писать функции, называемые обработчиками событий. Создавая обработчик того или и ого события, вы поручаете программе выполнить написанную функцию, если это событие произойдет

Для того, чтобы добавить обработчик событий, нужно выбрать на форме с помощью мыши компонент, которому необходим обработчик событий, затем открыть страницу событий инспектора объектов и дважды щелкнуть левой клавишей мыши на колонке значений рядом с событием, чтобы заставить C++ Builder сгенерировать прототип обработчика событий и показать его в редакторе кода. При этом автоматически генерируется текст пустой функции, и редактор открывается в том месте, где следует вводить код. Внутри операторных скобок {...} нужно ввести код, который должен выполняться при наступлении события. Обработчик событий может иметь параметры, которые указываются после имени функции в круглых скобках. (Эл. ресурс http://h-l-l.ru/publ/44)

Работа с файлами осуществляется при помощи встроенной в язык С++ функции fopen(), которая открывает файл с указанным именем и возвращает указатель типа FILE на этот файл. Функция реализует 6 типов доступа к файлу и 2 типа открытия файла относительно содержимого (бинарный или текстовый файл). После работы с файлом его обязательно необходимо закрывать при помощи встроенной функции fclose(). (Перевод с англ. http://www.cplusplus.com/reference/cstdio/fopen/)

**1.3 Алгоритм решения задачи**

Для начала формируем структуру, которая содержит поля, в совокупности формирующие базу данных.

struct Employee

{

char secondName[LENGTH\_STRING]; // second name of employye

char nameOfDep[LENGTH\_STRING]; // name of department

ProductByCategory doneProduct; // count of done product

} employee;

Поле secondName[LENGTH\_STRING] содержит фамилию рабочего. Поле nameOfDep[LENGTH\_STRING] содержит наименования цеха, к которому приписан рабочий. Поле doneProduct объявляем как объект другого структурного типа.

struct ProductByCategory

{

short int prodA; // A category of product

short int prodB; // B category of product

short int prodC; // C category of product

};

Структура содержащего 3 поля – количество собранных изделий 3-ех типов.

Так же объявляем третий тип данных, содержащий 3 ставки оплаты за собранные изделия 3 типов соответственно и создаем объект данного типа.

struct

{

short int salA; // salary for product A

short int salB; // salary for product B

short int salC; // salary for product C

}

Сразу же создаем объект типа Employee для последующей записи, чтения и обработки.

В событии создания главной формы TForm1::FormCreate() при помощи функции system() меняем кодовую страницу для возможности отображения кириллицы, очищаем все поля ввода и таблицу StringGrid1, устанавливаем свойства таблицы, изменяющие количество и размеры элементов таблицы, устанавливаем запрет на ее изменение, вызываем функцию печати базы данных из файла, имя которого также задано по умолчанию.

Событие TForm1::BitBtn2Click() создает новую запись. Другими словами, программа считывает информацию из заранее заполненных полей, расположенных на первой странице «Работа с базой данных» объекта PageControl1, записывает ее в объект структурного типа, который в свою очередь записывается в файл. Такая запись объектами структурного типа позволяет упорядочено хранить данные в файле и при аналогичном чтении из файла гарантирует безопасное извлечение информации. Затем поля ввода очищаются, таблица отображения базы данных обновляется новой записью.

Событие TForm1::BitBtn4Click() устанавливает 3 ставки оплаты за собранные изделия 3-ех типов соответственно, считывая информацию из полей ввода, расположенных на 2-ой странице «Задать ставки» объекта PageControl1. После успешной установки делает активной 3 страницу “Печать ведомостей” объекта PageControl1

Событие TForm1::BitBtn3Click() очищает объект Memo1 и передает управление 3-ем функциям, выдающим на печать ведомости различного содержания.

Также для решения поставленной задачи нам нужно создать меню, в которое будут задействованы основные компоненты программы. Меню задается с помощью компоненты MainMenu и включает в себя следующие пункты:

* Файл
  + Создать
  + Открыть
  + Очистить текущий
* Справка
  + О программе
  + Об авторе
* Выход

Описание каждого из пунктов меню:

1. Создать. Открывается форма, в поле которой нужно ввести название нового файла без расширения, которое предопределено (.txt) и изменить можно только программно, но нежелательно.
2. Открыть. При помощи компоненты OpenDialog открывается стандартное окно Windows открытия файла.
3. Очистить текущий. Вызывает соответствующий метод-событие, полностью очищающий файл, который обрабатывается в данных момент.
4. О программе. Открывает форму с описанием и возможностями программы.
5. Об авторе. Открывает форму с информацией об авторе программы.
6. Выход. Вызывает немедленное закрытие программы.

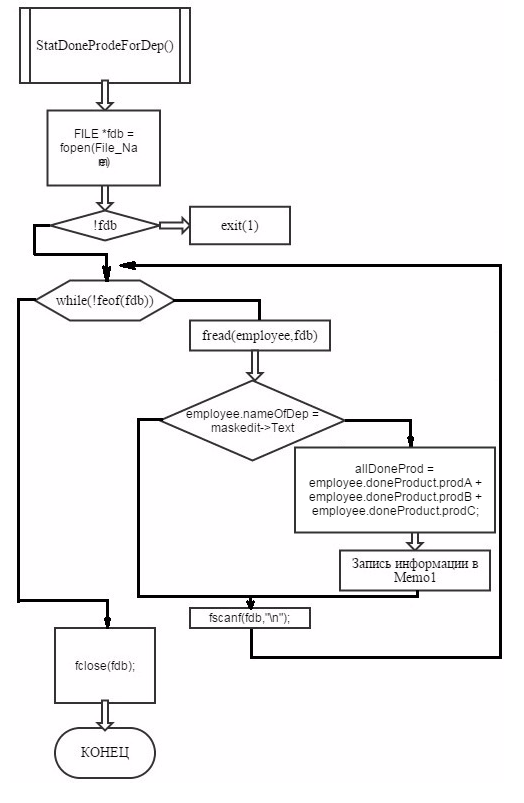
**1.4 Блок-схема алгоритма решения задачи**

Блок-схема организации создания базы данных и печати ведомостей представлена на рисунке 1.1



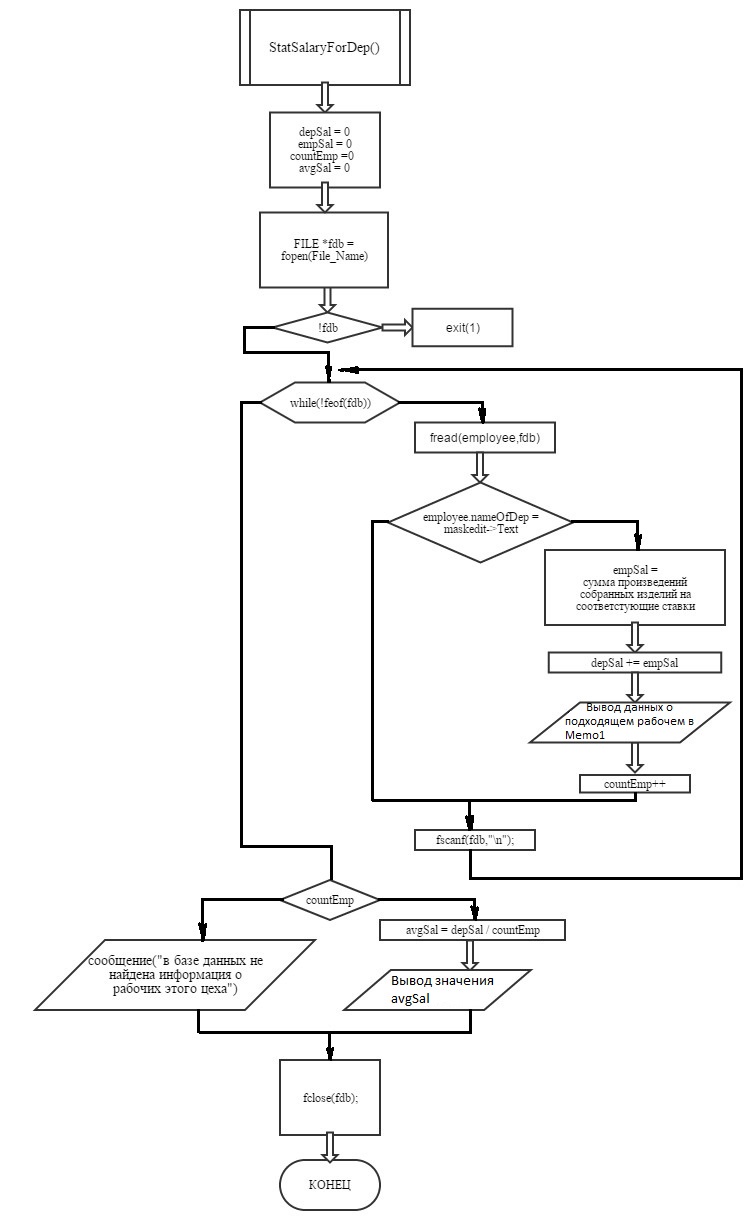
**Рисунок 1.1-Основная блок-схема**

Блок-схема функции, организующей печать ведомости «общее количество изделий категорий А, В, С, собранных рабочим цеха Х» показана на рисунке 1.2



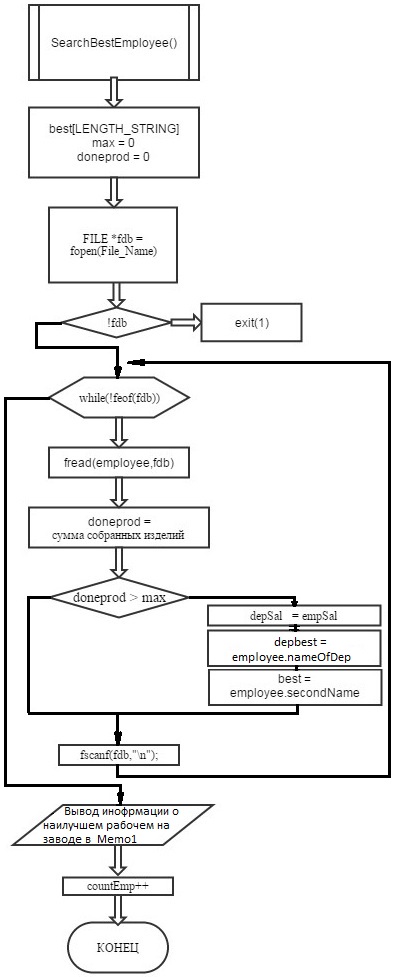
**Рисунок 1.2- Ведомость «общее количество изделий категорий А, В, С, собранных рабочим цеха Х»**

Функция, которая выдает на печать ведомость «ведомость зарплаты рабочих цеха Х» и информацию о средней зарплате цеха, изображена на рисунке 1.3

****

**Рисунок 1.3-Вывод ведомости** **«ведомость зарплаты рабочих цеха Х» и информации о средней зарплате цеха**

Функция, которая выдает на печать фамилию рабочего завода, собравшего наибольшее количество изделий, изображена на рисунке 1.4

****

**Рисунок 1.4-Вывод фамилии самого продуктивного рабочего.**

**2.ПРАКТИЧЕСКАЯ ЧАСТЬ**

**2.1Описание объектов, свойств, методов**

**Компонент BitBtn -** кнопка c возможностью добавлять картинку к тексту, наиболее распространенная в приложениях. Основные свойства представлены в таблице 1.1

Таблица 1.1-Свойства компонента BitBtn

|  |  |
| --- | --- |
| **Свойство** | **Назначение** |
| **Caption** | Надпись на компоненте. |
| **Color** | Цвет компонента. |
| **Glyph** | Открывает окно выбора картинки, после выбора которой будет отображаться на кнопке |
| **Enabled** | Если это свойство установлено в **false**, то компонент становится бледнее и по нему уже нельзя щелкнуть клавишей мыши (можно, но действие в обработчике щелчка не произойдет) или ввести/скопировать какие-либо данные. |
| **Font** | Шрифт компонента.    **Name** - Имя шрифта |
| **Height** | Высота компонента |
| **Visible** | Если установлено **true**, то компонент становится невидимым. |
| **Width** | Ширина компонента |

**Компонент StringGrid-**таблица, ячейки которой могут содержать текстовую информацию. Благодаря множеству свойств объект можно настроить очень разнообразно как в визуальном аспекте, так и в интерактивном. Основные свойства представлены в таблице 1.2

Таблица 1.2-Свойства компонента StringGrid

|  |  |
| --- | --- |
| **Свойство** | **Назначение** |
| **FixedCols** | Количество фиксированных столбцов, которые зачастую отображают фундаментальную информацию, например нумерацию списка. |
| **FixedRows** | Количество фиксированных строк, которые зачастую отображают фундаментальную информацию, например тип данных, хранящихся в столбце под ячейкой фиксированной строки. |
| **Enabled** | Если это свойство установлено в **false**, то компонент становится бледнее и по нему уже нельзя щелкнуть клавишей мыши (можно, но действие в обработчике щелчка не произойдет). |
| **Font** | Шрифт компонента.    **Name** - Имя шрифта |
| **Height** | Высота компонента |
| **RowCount / ColCount** | Количество строк / столбцов. |
| **Width** | Ширина компонента |

Так же в процессе работы программы опция **goEditing** изменяется с **true** на **false** и обратно в зависимости от того, требуется ли запрет на изменение данных таблицы или нет.

**Компонент MaskEdit.** Поле для ввода с установкой маски ввода – формата, которого следует придерживаться. Основные свойства представлены в таблице 1.3.

Таблица 1.3-Свойства компонента MaskEdit

|  |  |
| --- | --- |
| **Свойство** | **Назначение** |
| **Visible** | Если установлено **true**, то компонент становится невидимым. |
| **Text** | Текст, отображаемый в поле ввода. |
| **Enabled** | Если это свойство установлено в **false**, то компонент становится бледнее и по нему уже нельзя щелкнуть клавишей мыши (можно, но действие в обработчике щелчка не произойдет). |
| **Font** | Шрифт компонента.    **Name** - Имя шрифта |

**Компонент MainMenu.** Основное меню, которое появляется в верхнем левом углу. Основные свойства представлены в таблице 1.4.

Таблица 1.4-Свойства компонента MainMenu

|  |  |
| --- | --- |
| **Свойство** | **Значение** |
| **Bitmap** | Позволяет загрузить изображение в формате **bmp** для отображения у пункта меню |
| **Caption** | Отвечает за название пункта меню |
| **Enabled** | Определяет возможность нажатия на пункт меню, если установлено в **false**, то пользователь не сможет его выбрать(оно будет затемненным) |
| **Name** | Имя пункта меню. Представляется в виде **N1**, **N2**, **N3** и так далее. Рекомендуется переименовывать для облегчения работы с ним в будущем. |
| **ShortCut** | Определяет "горячую клавишу" или сочетание клавиш для быстрого вызова данного пункта меню |

**Компонент Memo.** Основное меню, которое появляется в верхнем левом углу. Основные свойства представлены в таблице 1.5.

Таблица 1.5-Свойства компонента Memo

|  |  |
| --- | --- |
| **Свойство** | **Значение** |
| **Font** | Шрифт компонента.    **Name** - Имя шрифта. |
| **ScrollBars** | Установка полос прокрутки |
| **Lines** | Содержимое объекта |
| **ReadOnly** | **true/false –** запрет/разрешение на изменение содержимого. |

**Компонент PageControl.** Позволяет объединять на одной форме несколько вкладок, которые содержат разные элементы управления

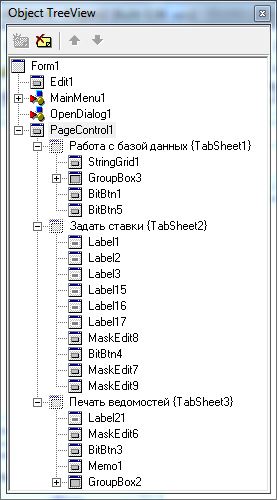
**Компонент TabSheet.** Является дочерним для объекта PageControl и представялет собой его вкладку.

**Компонент OpenDialog.** Компонент служит для открытия файлов.

**Компонент GroupBox.** Объединяет элементы, которые необходимо визуально сгруппировать.

**2.2 Формы и основные модули текста программы**

При решении поставленной задачи будут использованы объекты, изображенные на рисунке 2.1.



**Рисунок 2.1 - Объекты главной формы(Form1)**

Edit1 – печать имени обрабатываемого файла;

MainMenu1 - основное меню, которое служит для выбора основных настроек программы;

OpenDialog1 - компонента для открытия файла;

StringGrid1 – отображает записи базы данных;

BitBtn1 – разрешает изменение записей;

BitBtn5 - сохраняет изменения и запрещает корректировку данных;

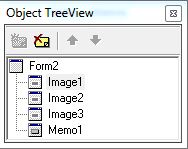
MaskEdit7..9 – поля для ввода 3-ех ставок соответсвенно;

BitBtn4 – сохраняет установку ставок;

MaskEdit6 – поле ввода наименования цеха, информацию о котором вы хотите видеть в ведомостях;

Memo1 – объект для печати ведомостей.

Основные компоненты Form2 показаны на рисунке 2.2

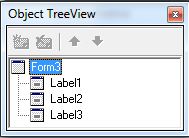
****

**Рисунок 2.2 - Объекты дополнительной формы(Form2)**

Memo - поле, предназначенное для вывода информации функциях и возможностях программы;

Image1..3 – компоненты, содержащие изображения работы программы.

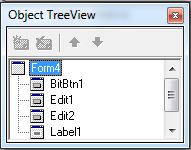
Основные компоненты Form3 показаны на рисунке 2.3



**Рисунок 2.3 - Объекты дополнительной формы(Form3)**

Label1..3 - компоненты, содержащие информацию об авторе программы.

Основные компоненты Form3 показаны на рисунке 2.4

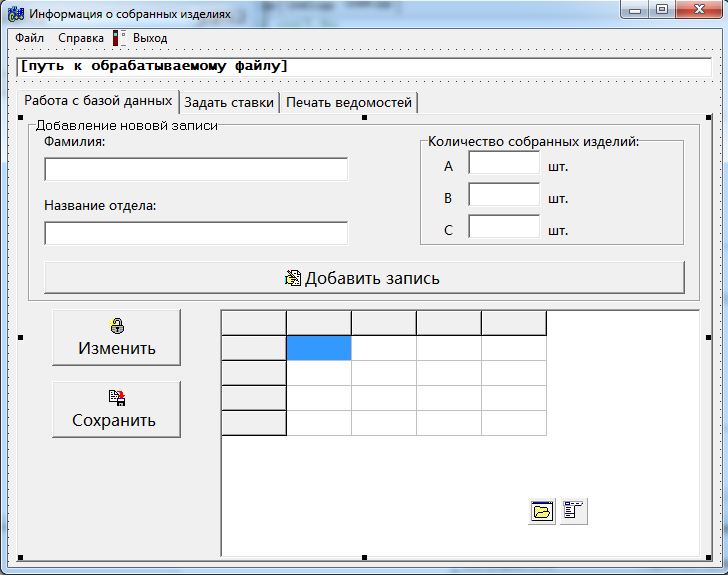


**Рисунок 2.4 - Объекты дополнительной формы(Form4)**

Edit1 – поле ввода имени нового файла;

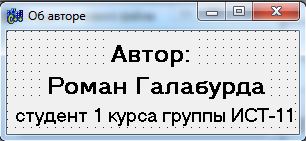
Edit2 – поле, отображающее предопределенное расширение файла;

BitBtn1 – вызывает событие создания нового файла с введенным именем.

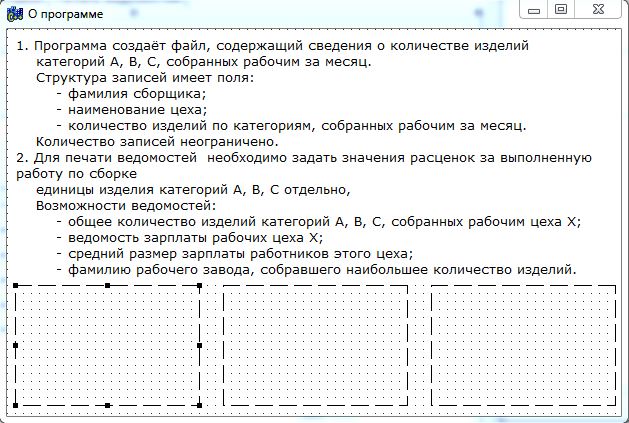
Основная форма приложения имеет вид, изображенный на рисунке 2.5

**Рисунок 2.5 - Вид основной формы программы**

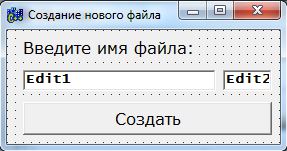
Дополнительные формы показаны на рисунках 2.6, 2.7 и 2.8



**Рисунок 2.6 - Вид дополнительной формы программы(Form2)**



**Рисунок 2.7 - Вид дополнительной формы программы(Form3)**



**Рисунок 2.8 - Вид дополнительной формы программы(Form4)**

**Основной модуль текста программы:**

В данном фрагменте кода программы представлены 3 функции, которые реализуют обработку файла-базы данных и печать ведомостей.

void StatDoneProdForDep(TMaskEdit \*maskedit, TMemo \*memo, char \*file\_name)

{

FILE \*fdb = fopen(file\_name,"rt");

if(!fdb) { ShowMessage("Файл отсутствует!"); perror(File\_Name); exit(1); }

// list with count of all done product for each employee

Form1->Memo1->Lines->Add(AnsiString("Работники цеха ") + maskedit->Text + AnsiString(":"));

while(!feof(fdb))

{

fread(&employee, sizeof(Employee), 1, fdb);

if(!strcmp(employee.nameOfDep,maskedit->Text.c\_str()))

{

int allDoneProd = employee.doneProduct.prodA +

employee.doneProduct.prodB +

employee.doneProduct.prodC;

memo->Lines->Add(AnsiString(" ")+employee.secondName +

AnsiString(" всего собрал ") +

IntToStr(allDoneProd) +

AnsiString(" изделий: (A: ") +

IntToStr(employee.doneProduct.prodA) +

AnsiString(", B: ") +

IntToStr(employee.doneProduct.prodB) +

AnsiString(", C: ") +

IntToStr(employee.doneProduct.prodC) +

AnsiString(")."));

}

fscanf(fdb,"\n");

}

fclose(fdb);

}

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

void StatSalaryForDep(TMaskEdit \*maskedit, TMemo \*memo, char \*file\_name )

{

int depSal = 0, // salary for interesting department

empSal = 0, // salary for employee from this department

countEmp = 0; // count of employees

double avgSal = 0; // average salary for interesting department

// calculate average salary for interesting department

// list with salary for each employee from this department

memo->Lines->Add(AnsiString("Оклад работников цеха ") +

maskedit->Text + AnsiString(":"));

FILE \*fdb = fopen(file\_name,"rt");

if(!fdb) { ShowMessage("Файл отсутствует!"); perror(File\_Name); exit(1); }

while(!feof(fdb))

{

fread(&employee, sizeof(Employee), 1, fdb);

if(!strcmp(employee.nameOfDep,maskedit->Text.c\_str()))

{

empSal = employee.doneProduct.prodA \* salary.salA +

employee.doneProduct.prodB \* salary.salB +

employee.doneProduct.prodC \* salary.salC;

depSal += empSal;

memo->Lines->Add(AnsiString(" Оклад работника ") +

employee.secondName + AnsiString(" составляет ") + IntToStr(empSal) + AnsiString("$"));

countEmp++;

}

fscanf(fdb,"\n");

}

if(countEmp)

{

avgSal = depSal / countEmp;

Form1->Memo1->Lines->Add(AnsiString("Средний оклад по цеху ") + maskedit->Text + AnsiString(":") + FormatFloat(0.00, avgSal) + "$");

}

else

ShowMessage("В базе данных не найдена информация о сотрудниках искомого цеха!");

fclose(fdb);

}

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

void SearchBestEmployee()

{

FILE \*fdb = fopen(File\_Name,"rt");

if(!fdb) { perror(File\_Name); exit(1); }

char best[LENGTH\_STRING] , depbest[LENGTH\_STRING];

int max = 0, doneprod = 0;

while(!feof(fdb))

{

fread(&employee, sizeof(Employee), 1, fdb);

doneprod = employee.doneProduct.prodA +

employee.doneProduct.prodB +

employee.doneProduct.prodC;

if(doneprod > max)

{

max = doneprod;

strcpy(best,employee.secondName);

strcpy(depbest,employee.nameOfDep);

}

fscanf(fdb,"\n");

};

Form1->Memo1->Lines->Add("Лучший работник завода:");

Form1->Memo1->Lines->Add(AnsiString("Рабочий ") + best + AnsiString(" цеха ") + depbest + " собрал больше всего изделий: " + IntToStr(max) +

AnsiString(" шт."));

fclose(fdb);

}

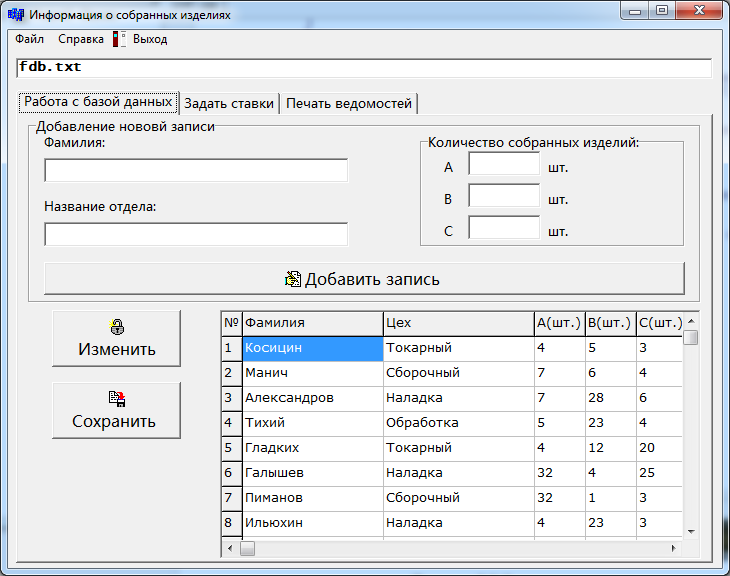
**2.3 Результаты тестирования программного продукта**

Краткие сведения о программе:

1. Код основной формы находится в **Unit1.сpp**
2. При запуске приложения вверху главной формы мы увидим имя текущего обрабатываемого файла.
3. Выбор другого файла осуществляется при нажатии на вкладку меню “Файл -> Создать” для создания нового файла или “Файл -> Открыть” для открытия уже существующего файла.
4. В таблице мы увидим существующие записи о рабочих.
5. Для изменения записей требуется нажать кнопку «Изменить», откорректировать данные в таблице и для сохранения изменений и перезаписи файла нажать кнопку «Сохранить»
6. Для добавления новой записи нужно заполнить все поля ввода, находящиеся на первой вкладке, и нажать кнопку «Добавить запись». Таблица автоматически обновится и дополнится новой записью.
7. При нажатии на пункт меню “Файл -> Очистить текущий” вы можете очистить файл, обрабатываемый в данный момент работы программы.
8. Для показа ведомостей в первую очередь необходимо установить ставки за собранные изделия. Сделать это можно на второй вкладке программы: заполните 3 поля, нажмите кнопку «Установить». При корректном вводе данных появится окно с сообщением успешной установке и программы автоматически сделает активной 3-ю вкладку.
9. Для печати ведомостей необходимо нажать кнопку «Показать»
10. Код второй формы находится в **Unit2.cpp**
11. Код третьей формы находится в **Unit3.cpp**
12. Код четвертой формы находится в **Unit4.cpp**
13. При нажатии на кнопку “Создать” создастся новый файл и форма закроется.

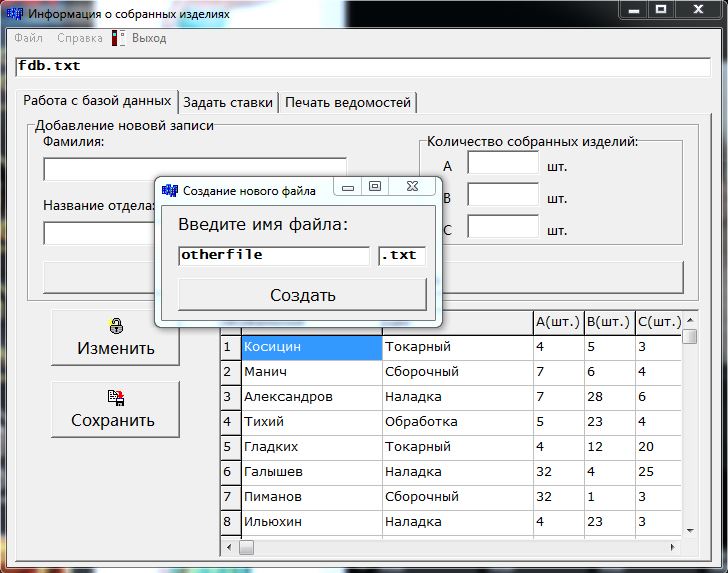
Для открытия программы нужно запустить файл **Project1. exe.**

По умолчанию открывается первая вкладка программы, отображающая интерфейс добавления новых записей и таблицу уже существующей базы данных, которая находится в файла, имя которого также предустановлено. Для выбора другого файла или создания нового можно воспользоваться пунктом меню «Файл». Первая вкладки программы изображена на 2.9



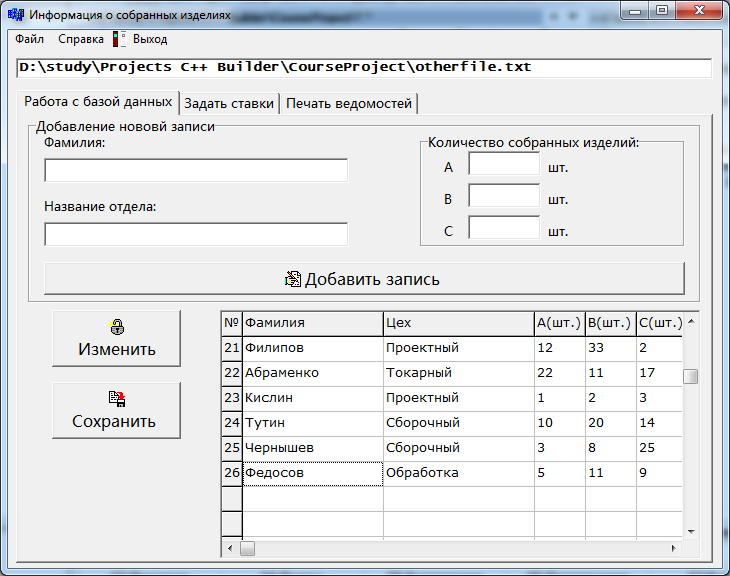
**Рисунок 2.9-Первая вкладка программы**

Форма создания файла изображена на рисунке 2.10



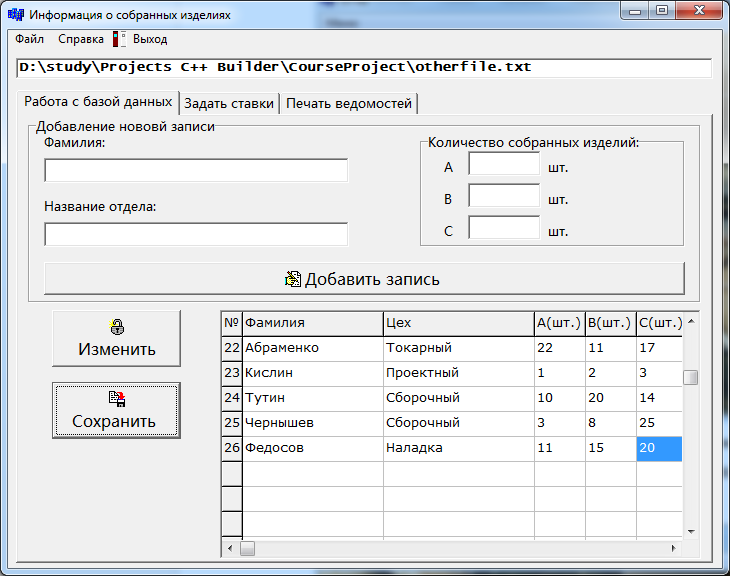
**Рисунок 2.10-Форма для создания нового файла**

Добавили запись, таблица обновилась и новая запись отобразилась под номером 26. Форма примет вид, изображенный на рисунке 2.11



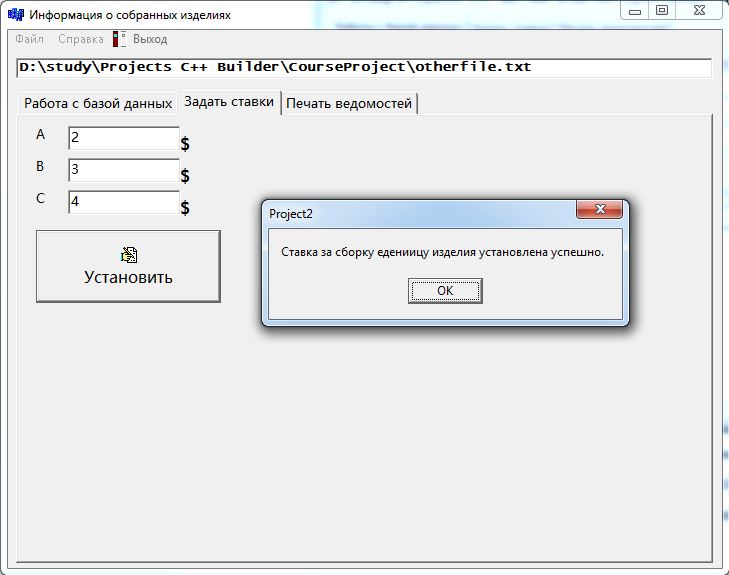
**Рисунок 2.11-После добавления записи**

Нажмем кнопку «Изменить» и изменим данные только что добавленной записи. Затем нажмем «Сохранить». Форма примет вид, изображенный на рисунке 2.12



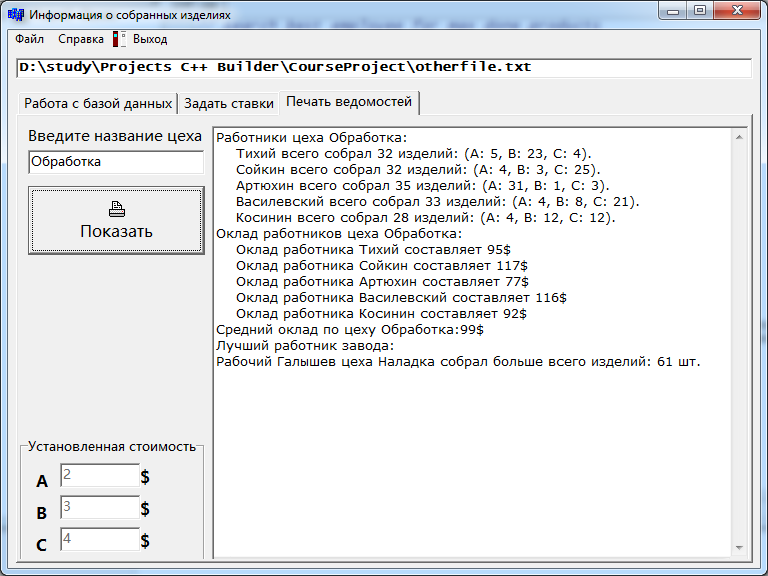
**Рисунок 2.12-После изменения данных**

Далее перейдем на вторую вкладки программы «Задать ставки». Заполним поля и нажмем кнопку «Установить». Появится окно с сообщением об успешной установке ставок и автоматически активной станет третья вкладка программы «Печать ведомостей». Результат выполнения данной команды показан на рисунке 2.13



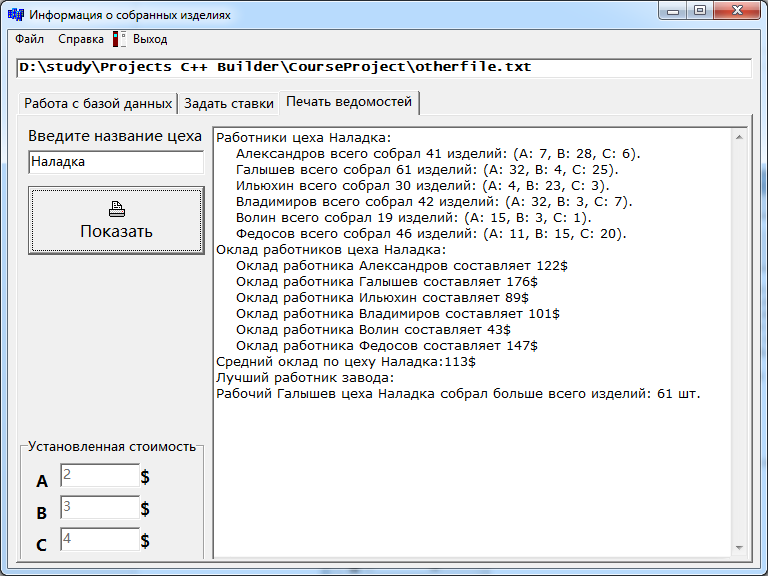
**Рисунок 2.13-Установка ставок выполнена успешно**

В левом нижнем углу вкладки «Печать ведомостей» можно увидеть информацию и текущих значениях ставок. Теперь для получения информации о цехе «Обработка» необходимо ввести название цеха в поле и нажать кнопку «Показать». Предположим, что ставки не установлены, но пользователь нажимает кнопку «Показать». В этом случае появится сообщение с предупреждением о том, что ставки не заданы. Но так как ставки установлены, то информационное поле заполняется ведомостями о введенном цехе. Программы примет вид, изображенный на рисунке 2.14



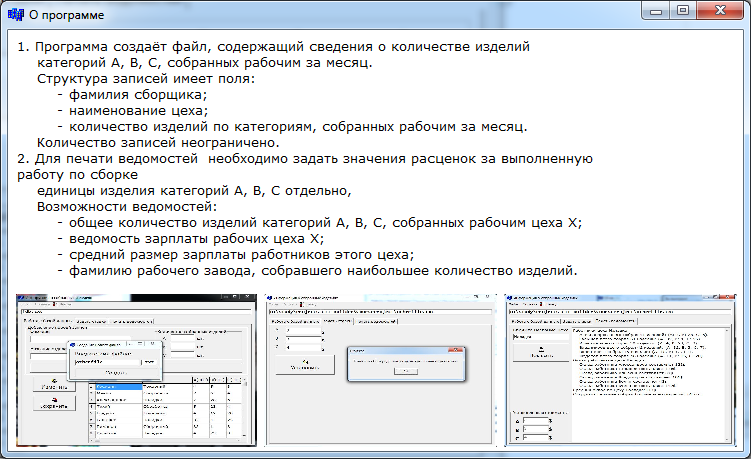
**Рисунок 2.14-Ведомость по цеху «Обработка»**

Чтобы убедиться в том, что программы учитывает добавленного нами рабочего под номером 26, в поле названия цеха необходимо ввести «Наладка». В результате печатается ведомость, изображенная на рисунке 2.15



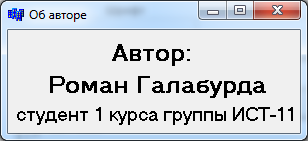
**Рисунок 2.15- Ведомость по цеху «Наладка»**

При нажатии на пункт меню «Справка->О программе» открывается форма с информацией о функциях и возможностях программы, изображенная на рисунке 2.16



**Рисунок 2.16-Форма «О программе»**

При нажатии на пункт меню «Справка->Об авторе» открывается форма с информацией об авторе, изображенная на рисунке 2.17



**Рисунок 2.16-Форма «Об авторе»**

**2.4 Анализ результатов**

Результаты тестирования программы показывают, что приложение работает корректно. Выполнение программы соответствует всем пунктам поставленной задачи. При запуске приложения все компоненты программы работают правильно.

# 

**ЗАКЛЮЧЕНИЕ**

Данная работа показывает, что процесс разработки настольного бизнес-приложения требует немалого количества знаний о языке программирования С++ и возможностях среды разработки С++ Builder.

Кроме того, важными моментами при разработки программы являются описание спецификации программы, алгоритма ее реализации, хотя C++ и предоставляет необходимые средства для упрощения процесса программирования.

Проведенное тестирование программы позволяет сделать вывод о работоспособности программы и ее соответствии заданию. Как и любое решение в любой области наук, промышленности или социальной жизни, так и любую программу можно усовершенствовать. То есть дополнять новыми возможностями, делать интерфейс более интерактивным и дружелюбным, разрабатывать дизайн приложения и т.д. С увеличением опыта и навыков программист может разрабатывать приложение намного сложнее и объемнее.

А так как области, в которых данное приложение может найти применение, развиваются все больше и больше, то разработка подобных приложений все более востребована, а тем более разработка объемных в плане своих возможностей приложений.

**СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ**

1. Архангельский, А. Я. Программирование в C++Builder / А. Я. Архангельский – Москва: ООО "Бином-Пресс", 2007. – 1184 с.
2. Information on the C++ language - 2000 Режим доступа: www.cplusplus.com
3. Компоненты C++ Builder // High Level Language C++ [Электронный ресурс]. – 2014. – Режим доступа: http://h-l-l.ru/publ/44– Дата доступа: 10.03.2014.
4. Б. Керниган, Д. Ричи. «Язык программирования Си» [Электронный ресурс]. – 2012. Режим доступа: http://people.toiit.sgu.ru/Sinelnikov/PT/C/Kern\_Ritch.pdf Дата доступа: 03.04.2014.
5. Windows-1251 [Электронный ресурс]. – 2014. Режим доступа: https://ru.wikipedia.org/wiki/Windows-1251 Дата доступа: 03.04.2014.

# ПРИЛОЖЕНИЕ А

# Код программы

# #define \_CRT\_SECURE\_NO\_WARNINGS

# #include <vcl.h>

# #include <stdio.h>

# #include <stdlib.h>

# #pragma hdrstop

# #include "Unit1.h"

# #include "Unit2.h"

# #include "Unit3.h"

# #include "Unit4.h"

# #pragma package(smart\_init)

# #pragma resource "\*.dfm"

# TForm1 \*Form1;

# \_\_fastcall TForm1::TForm1(TComponent\* Owner) : TForm(Owner) {}

# #define LENGTH\_STRING 16

# char \*File\_Name = {"fdb.txt"};

# int CountEmp = 0;

# void StatDoneProdForDep(TMaskEdit \*maskedit, TMemo \*memo, char \*file\_name);

# void StatSalaryForDep(TMaskEdit \*maskedit, TMemo \*memo, char \*file\_name);

# void SearchBestEmployee();

# void PrintDBInStringGrid(TStringGrid \*grid, char \*file\_name);

# struct ProductByCategory

# {

# short int prodA; // A category of product

# short int prodB; // B category of product

# short int prodC; // C category of product

# };

# struct Employee

# {

# char secondName[LENGTH\_STRING]; // second name of employye

# char nameOfDep[LENGTH\_STRING]; // name of department

# ProductByCategory doneProduct; // count of done product

# } employee;

# struct

# {

# short int salA; // salary for product A

# short int salB; // salary for product B

# short int salC; // salary for product C

# } salary = {0};

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

# {

# system("chcp 1251 > nul");

# PageControl1->ActivePageIndex = 0;

# MaskEdit7->Text = IntToStr(0);

# MaskEdit8->Text = IntToStr(0);

# MaskEdit9->Text = IntToStr(0);

# File\_Name = "fdb.txt";

# Edit1->Text = File\_Name;

# StringGrid1->ColCount = 6;

# StringGrid1->RowCount = 100;

# StringGrid1->ColWidths[0] = 20;

# StringGrid1->ColWidths[1] = 140;

# StringGrid1->ColWidths[2] = 150;

# StringGrid1->ColWidths[3] = 50;

# StringGrid1->ColWidths[4] = 50;

# StringGrid1->ColWidths[5] = 50;

# StringGrid1->Cells[0][0] = "№";

# StringGrid1->Cells[1][0] = "Фамилия";

# StringGrid1->Cells[2][0] = "Цех";

# StringGrid1->Cells[3][0] = "À(шт.)";

# StringGrid1->Cells[4][0] = "B(шт.)";

# StringGrid1->Cells[5][0] = "C(шт.)";

# StringGrid1->Options = StringGrid1->Options >> goEditing;

# PrintDBInStringGrid(StringGrid1, File\_Name);

# }

# // create new embloyee

# void \_\_fastcall TForm1::BitBtn2Click(TObject \*Sender)

# {

# FILE \*fdb = fopen(File\_Name,"at");

# if(!fdb) { perror(File\_Name); exit(1); }

# // read information from form and save him in file

# strcpy(employee.secondName,MaskEdit1->Text.c\_str()); // second name

# strcpy(employee.nameOfDep,MaskEdit2->Text.c\_str()); // department

# employee.doneProduct.prodA = StrToInt(MaskEdit3->Text); // product A

# employee.doneProduct.prodB = StrToInt(MaskEdit4->Text); // product B

# employee.doneProduct.prodC = StrToInt(MaskEdit5->Text); // product C

# MaskEdit1->Text = "";

# MaskEdit2->Text = "";

# MaskEdit3->Text = "";

# MaskEdit4->Text = "";

# MaskEdit5->Text = "";

# fwrite(&employee, sizeof(Employee), 1, fdb);

# fprintf(fdb,"\n");

# fclose(fdb);

# PrintDBInStringGrid(StringGrid1, File\_Name);

# }

# // set salary for single done product

# void \_\_fastcall TForm1::BitBtn4Click(TObject \*Sender)

# {

# salary.salA = StrToInt(MaskEdit7->Text);

# salary.salB = StrToInt(MaskEdit8->Text);

# salary.salC = StrToInt(MaskEdit9->Text);

# if((salary.salA) && (salary.salB) && (salary.salC))

# {

# ShowMessage("Ставка за сборку единицы изделия установлена успешно.");

# MaskEdit10->Text = MaskEdit7->Text;

# MaskEdit11->Text = MaskEdit8->Text;

# MaskEdit12->Text = MaskEdit9->Text;

# }

# else

# ShowMessage("Во время задания данных возникла ошибка! Попробуйте еще раз.");

# TabSheet3->Show();

# }

# // print reports

# void \_\_fastcall TForm1::BitBtn3Click(TObject \*Sender)

# {

# Form1->Memo1->Clear();

# StatDoneProdForDep(MaskEdit6 ,Memo1, File\_Name);

# StatSalaryForDep(MaskEdit6 ,Memo1, File\_Name);

# SearchBestEmployee();

# }

# // change database

# void \_\_fastcall TForm1::BitBtn1Click(TObject \*Sender)

# { StringGrid1->Options = StringGrid1->Options << goEditing;}

# // save (read from StringGrid1 and rewrite file)

# void \_\_fastcall TForm1::BitBtn5Click(TObject \*Sender)

# {

# StringGrid1->Options = StringGrid1->Options >> goEditing;

# FILE \*fdb = fopen(File\_Name,"wt");

# if(!fdb) { perror(File\_Name); exit(1); }

# int count = 1;

# while(count <= CountEmp)

# {

# strcpy(employee.secondName, StringGrid1->Cells[1][count].c\_str());

# strcpy(employee.nameOfDep, StringGrid1->Cells[2][count].c\_str());

# employee.doneProduct.prodA = StrToInt(StringGrid1->Cells[3][count]);

# employee.doneProduct.prodB = StrToInt(StringGrid1->Cells[4][count]);

# employee.doneProduct.prodC = StrToInt(StringGrid1->Cells[5][count]);

# fwrite(&employee, sizeof(Employee), 1, fdb);

# fprintf(fdb,"\n");

# count++;

# }

# fclose(fdb);

# PrintDBInStringGrid(StringGrid1, File\_Name);

# }

# // clear Edit for enter data

# void \_\_fastcall TForm1::MaskEdit7Click(TObject \*Sender)

# { MaskEdit7->Text = ""; }

# void \_\_fastcall TForm1::MaskEdit8Click(TObject \*Sender)

# { MaskEdit8->Text = ""; }

# void \_\_fastcall TForm1::MaskEdit9Click(TObject \*Sender)

# { MaskEdit9->Text = ""; }

# // open information about this program end exit of application

# void \_\_fastcall TForm1::N4Click(TObject \*Sender)

# { Application->Terminate(); }

# void \_\_fastcall TForm1::N2Click(TObject \*Sender)

# { Form2->Show(); }

# void \_\_fastcall TForm1::N3Click(TObject \*Sender)

# { Form3->Show(); }

# // open file

# void \_\_fastcall TForm1::N7Click(TObject \*Sender)

# {

# if(OpenDialog1->Execute())

# {

# File\_Name = OpenDialog1->FileName.c\_str();

# Edit1->Text = File\_Name;

# }

# else

# {

# File\_Name = "filedatabase.dat";

# Edit1->Text = File\_Name;

# }

# PrintDBInStringGrid(StringGrid1, File\_Name);

# }

# // clear file

# void \_\_fastcall TForm1::N8Click(TObject \*Sender)

# {

# FILE \*fdb = fopen(File\_Name,"wt");

# if(!fdb) { perror(File\_Name); exit(1); }

# ShowMessage(File\_Name + AnsiString(" : очистка прошла успешно!"));

# fclose(fdb);

# }

# // create new file

# void \_\_fastcall TForm1::N6Click(TObject \*Sender)

# { Form4->Show();}

# // report about all products made by employees from interesting department

# void StatDoneProdForDep(TMaskEdit \*maskedit, TMemo \*memo, char \*file\_name)

# {

# FILE \*fdb = fopen(file\_name,"rt");

# if(!fdb) { ShowMessage("Файл отсутствует!"); perror(File\_Name); exit(1); }

# // list with count of all done product for each employee

# Form1->Memo1->Lines->Add(AnsiString("работники цеха ") + maskedit->Text +

# AnsiString(":"));

# while(!feof(fdb))

# {

# fread(&employee, sizeof(Employee), 1, fdb);

# if(!strcmp(employee.nameOfDep,maskedit->Text.c\_str()))

# {

# int allDoneProd = employee.doneProduct.prodA +

# employee.doneProduct.prodB +employee.doneProduct.prodC;

# memo->Lines->Add(AnsiString(" ")+employee.secondName +

# AnsiString(" всего собрал ") +

# IntToStr(allDoneProd) +

# AnsiString(" изделий: (A: ") +

# IntToStr(employee.doneProduct.prodA) + AnsiString(", B: ") +

# IntToStr(employee.doneProduct.prodB) + AnsiString(", C: ") +

# IntToStr(employee.doneProduct.prodC) + AnsiString(")."));

# }

# fscanf(fdb,"\n");

# }

# fclose(fdb);

# }

# void StatSalaryForDep(TMaskEdit \*maskedit, TMemo \*memo, char \*file\_name )

# {

# int depSal = 0, // salary for interesting department

# empSal = 0, // salary for employee from this department

# countEmp = 0; // count of employees

# double avgSal = 0; // average salary for interesting department

# // calculate average salary for interesting department

# // list with salary for each employee from this department

# memo->Lines->Add(AnsiString("Оклад работников цеха ") +

# maskedit->Text + AnsiString(":"));

# FILE \*fdb = fopen(file\_name,"rt");

# if(!fdb) { ShowMessage("Файл отсутствует!"); perror(File\_Name); exit(1); }

# while(!feof(fdb))

# {

# fread(&employee, sizeof(Employee), 1, fdb);

# if(!strcmp(employee.nameOfDep,maskedit->Text.c\_str()))

# {

# empSal = employee.doneProduct.prodA \* salary.salA +

# employee.doneProduct.prodB \* salary.salB +

# employee.doneProduct.prodC \* salary.salC;

# depSal += empSal;

# memo->Lines->Add(AnsiString(" оклад работника ") + employee.secondName + AnsiString(" составляет ") + IntToStr(empSal) + AnsiString("$"));

# countEmp++;

# }

# fscanf(fdb,"\n");

# }

# if(countEmp)

# {

# avgSal = depSal / countEmp;

# Form1->Memo1->Lines->Add(AnsiString("Средний оклад по цеху ") +

# maskedit->Text + AnsiString(":") + FormatFloat(0.00, avgSal) + "$");

# }

# else

# ShowMessage("В базе данных не найдена информация о сотрудниках искомого цеха!");

# fclose(fdb);

# }

# // search best employee for max done products

# void SearchBestEmployee()

# {

# FILE \*fdb = fopen(File\_Name,"rt");

# if(!fdb) { perror(File\_Name); exit(1); }

# char best[LENGTH\_STRING] , depbest[LENGTH\_STRING];

# int max = 0, doneprod = 0;

# while(!feof(fdb))

# {

# fread(&employee, sizeof(Employee), 1, fdb);

# doneprod = employee.doneProduct.prodA +

# employee.doneProduct.prodB +

# employee.doneProduct.prodC;

# if(doneprod > max)

# {

# max = doneprod;

# strcpy(best,employee.secondName);

# strcpy(depbest,employee.nameOfDep);

# }

# fscanf(fdb,"\n");

# };

# Form1->Memo1->Lines->Add("Лучший работник завода:");

# Form1->Memo1->Lines->Add(AnsiString("Рабочий ") + best + AnsiString(" цеха ") + depbest + " собрал больше всего изделий: " + IntToStr(max) +

# AnsiString(" шт.")); AnsiString(" шт."));

# fclose(fdb);

# }

# // print database in StringGrid

# void PrintDBInStringGrid(TStringGrid \*grid, char \*file\_name)

# {

# grid->Options = grid->Options << goEditing;

# FILE \*fdb = fopen(file\_name, "rt");

# if(!file\_name) {perror(file\_name); exit(1);}

# int count = 1;

# while(!feof(fdb))

# {

# fread(&employee, sizeof(Employee),1,fdb);

# grid->Cells[0][count] = IntToStr(count);

# grid->Cells[1][count] = employee.secondName;

# grid->Cells[2][count] = employee.nameOfDep;

# grid->Cells[3][count] = employee.doneProduct.prodA;

# grid->Cells[4][count] = employee.doneProduct.prodB;

# grid->Cells[5][count] = employee.doneProduct.prodC;

# ++count;

# fscanf(fdb,"\n");

# }

# CountEmp = count - 1;

# fclose(fdb);

# grid->Options = grid->Options >> goEditing;

# }

# // safe enter of data

# void \_\_fastcall TForm1::MaskEdit1KeyPress(TObject \*Sender, char &Key){

# if (Key == VK\_BACK) return; if ((Key < 'À') || (Key > 'ß') && (Key < 'à') || ((Key > 'ÿ'))) Key = 0;}

# void \_\_fastcall TForm1::MaskEdit2KeyPress(TObject \*Sender, char &Key){

# if (Key == VK\_BACK) return; if ((Key < '0') || (Key > '9')) Key = 0;}

# ПРИЛОЖЕНИЕ Б

**Код второй формы**

#include <vcl.h>

#pragma hdrstop

#include "Unit2.h"

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TForm2 \*Form2;

\_\_fastcall TForm2::TForm2(TComponent\* Owner) : TForm(Owner){}

# ПРИЛОЖЕНИЕ В

**Код третьей формы**

#include <vcl.h>

#pragma hdrstop

#include "Unit3.h"

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TForm3 \*Form3;

\_\_fastcall TForm3::TForm3(TComponent\* Owner) : TForm(Owner) { }

# ПРИЛОЖЕНИЕ Г

**Код четвертой формы**

#include <vcl.h>

#include <stdio.h>

#pragma hdrstop

#include "Unit4.h"

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TForm4 \*Form4;

\_\_fastcall TForm4::TForm4(TComponent\* Owner) : TForm(Owner) { }

char \*\_File\_Name = "";

void \_\_fastcall TForm4::FormCreate(TObject \*Sender)

{

Edit1->Text = "";

Edit2->Text = ".txt";

Edit2->Enabled = "false";

}

void \_\_fastcall TForm4::BitBtn1Click(TObject \*Sender)

{

strcat(strcpy(\_File\_Name, Edit1->Text.c\_str()), ".txt");

ShowMessage(\_File\_Name);

FILE \*fdb = fopen(\_File\_Name,"wt");

if(!fdb) { perror(\_File\_Name); exit(1); }

fclose(fdb);

Form4->Close();

}