Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Оренбургский государственный университет»

Кафедра прикладной информатики в экономике и управлении

С.А. Вдович

ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ

Методические указания

Рекомендовано к изданию редакционно-издательским советом федерального государственного бюджетного образовательного учреждения высшего образования «Оренбургский государственный университет» для обучающихся по образовательным программам высшего образования по направлениям подготовки 38.03.05 Бизнес-информатика, 09.03.03 Прикладная информатика

Оренбург
2017

УДК 004.43
ББК 32.973.3
В27

Рецензент – доцент, кандидат экономических наук М.М. Пирязев

Вдович, С.А.
В27 Объектно-ориентированное программирование: методические указания / С.А. Вдович; Оренбургский гос. ун-т. – Оренбург: ОГУ, 2017. – 58 с.

Основное содержание: краткие сведения по основам объектно-ориентированного программирования в инструментальной среде разработки C++ Builder, показаны практические примеры реализации программ, даны методические рекомендации по выполнению лабораторных работ и задания на лабораторные работы.

Методические указания по дисциплинам «Объектно-ориентированное программирование», «Разработка программных приложений» предназначены для обучающихся по направлениям 38.03.05 Бизнес-информатика, 09.03.03 Прикладная информатика.

УДК 004.43
ББК 32.973.3

Содержание

Введение

Одной из технологий программирования является технология объектно-ориентированного программирования (ООП), которая основана на моделировании реального мира, при котором детали его реализации скрыты. ООП основано на трех концепциях: инкапсуляции, наследовании, полиморфизме. Основными понятиями ООП являются понятия класса и объекта. Класс представляет собой тип, создаваемый пользователем. Объект или экземпляр класса является переменной некоторого типа, определенного пользователем. Таким образом, класс это объектно-ориентированный инструмент для создания новых типов данных, являющихся объектами.

Данные методические указания к лабораторным работам предназначены для укрепления теоретического материала и освоения практических навыков по разработке программного обеспечения для решения экономических и расчетных задач с применением современных методов и технологий программирования. В качестве инструмента программирования предлагается среда визуального программирования Borland C++ Builder.

В методических указаниях представлено 6 лабораторных работ, охватывающих весь учебный курс, ориентированных на формирование у обучающихся следующих компетенций: ПК-2 способность разрабатывать, внедрять и адаптировать прикладное программное обеспечение; ПК-8 способность программировать приложения и создавать программные прототипы решения прикладных задач. Каждой лабораторной работе предшествует теоретический материал, пример реализации задач, варианты заданий для самостоятельного решения и контрольные вопросы по рассматриваемой теме. Также предлагаются рекомендации по выполнению и оформлению лабораторных работ и список рекомендуемой литературы.

1 Лабораторная работа №1. Объектно-ориентированный подход к проектированию и разработке программ

Цель работы: введение в ООП. Начальное ознакомление с инструментальной средой разработки (ИСР) Borland C++ Builder.

Краткие теоретические сведения

Объектно-ориентированное программирование (ООП) – это одна из технологий программирования, в которой используются объекты, а не алгоритмы. Основными принципами ООП являются наследие, инкапсуляция и полиморфизм и др. Эти принципы реализованы в программных языках высокого уровня, таких как C, C++, Паскаль, Java. Существует множеств инструментальных средств разработки программ, ориентированных на определенный программный язык. Для объектно-ориентированного программирования на C++ можно выделить наиболее популярные среды разработки Visual Studio C++ и Borland C++ Builder.

C++ Builder - программный продукт, инструмент быстрой разработки приложений (RAD), система, используемая программистами для разработки программного обеспечения на языке C++. Программный продукт принадлежит компании Embarcadero Technologies. Borland C++ Builder объединяет в себе комплекс объектных библиотек, таких как STL, VCL, CLX, MFC и др., компилятор, отладчик, редактор кода и многие другие компоненты.

Главное окно среды программирования состоит из главного меню, панели инструментов, палитры компонент (библиотека VCL). Палитра компонент оперирует визуальными элементами управления и содержит библиотеку из более 100 компонент, разделена карточными вкладками на несколько функциональных групп:

1) Standard - стандартная, содержащая наиболее часто используемые компоненты

2) Additional - дополнительная,являющаяся дополнением к стандартной

3) Win32 - 32-битные компоненты в стиле Windows95/98 и NT

4) System - системная, содержащая такие компоненты, как таймеры, плееры и ряд других

5) Data Access - доступ к данным через Borland Database Engine

6) Data Controls - управление данными

7) Internet - Интернет, компоненты для приложений, работающих с Интернет

8) Decision Cube - многомерный анализ данных

9) Qreport - быстрая подготовка отчетов

10) Dialogs - страница стандартных диалогов

11) Win 3.1 - Windows 3.x, компоненты в стиле Windows 3.x

12) Samples - образцы, различные интересные, но не до конца документированные компоненты

13) ActiveX - активные элементы ActiveX

Также главное окно программы включает текстовый редактор (Code Editor), конструктор форм (Form), администратор проектов (Project Manager), инспектор объектов (Object Inspector), архив объектов (Object Repository), навигатор по объектам (Browser), конструктор меню (Menu Designer), встроенный отладчик, локальные меню, справочную систему. Главное окно среды программирования Borland C++ Builder представлено на рисунке 1.

Основой всех приложений Borland C++ Builder является форма. Ее можно понимать как типичное окно Windows. Форма является основой, на которой размещаются другие компоненты. Чтобы перенести компонент на форму, надо открыть соответствующую страницу VCL и указать курсором мыши необходимый компонент. Затем нужно сделать щелчок мышью в нужном месте формы.

Форма имеет те же свойства, что присущи другим окнам Windows. Она имеет управляющее меню в верхнем левом углу, полосу заголовка, кнопки развертывания, свертывания и закрытия окна в верхнем правом углу. Можно изменить вид окна,

убрав в нем какие-то кнопки или всю полосу заголовка, сделав его окном с неизменяемыми размерами и т.д.

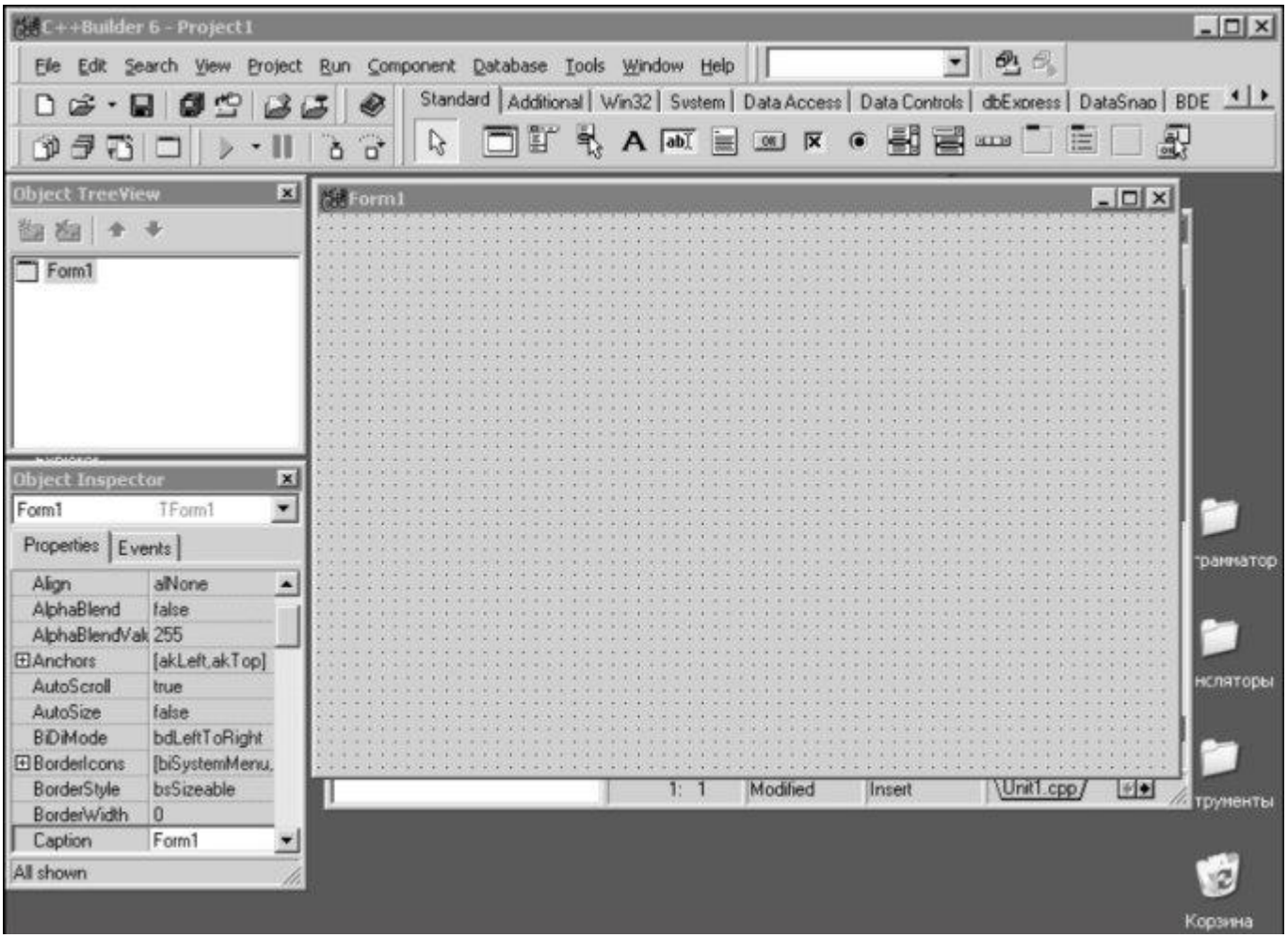

Рисунок 1 – Главное окно среды программирования Borland C++ Builder

Одной из наиболее важных частей среды Borland C++ Builder является окно Редактора Кода, или Текстового редактора. Он предназначен для размещения текстов программных единиц проектов с целью их создания, подключения к проекту, отладки, изменения (редактирования) или переименования. Отображается редактор в виде одного или нескольких окон. Каждое окно имеет одну или несколько страниц, причем на каждой странице находится текст отдельной программной единицы. На странице текстового редактора может также находиться текстовое представление формы - условное изображение ее параметров и параметров ее компонент. В нижней части окна Редактора Кода находится строка состояния, содержащая следующую информацию:

- расположение курсора редактора в тексте;

- признак изменения текста (Modified);

- режим работы с символами: вставки (Insert), замены (Overwrite), или только для чтения (Read only).

При создании новых программных единиц Borland C++ Builder формирует заготовки текстов этих программных единиц и помещает их в отдельные страницы активного окна редактора, а также устанавливает необходимые связи между компонентами проекта. Со своей стороны программист дополняет созданные заготовки конкретным содержанием.

В основном поле окна слева находится окно Инспектора Объектов. Он обеспечивает простой и удобный интерфейс для изменения свойств объектов Borland C++ Builder и управления событиями, на которые реагирует объект.

Окно Инспектора Объектов имеет две страницы. Выше их имеется выпадающий список всех компонентов, размещенных на форме. В нем вы можете выбрать тот компонент, свойства или события которого вас интересуют.

Страница свойств (Properties) Инспектора Объектов показывает свойства того объекта, который в данный момент выделен вами. Эти свойства можно изменять. На странице событий (Events) указаны все события, на которые может реагировать выбранный объект, с указанием имен обработчиков событий или отсутствием таковых. Для задания обработчика конкретного события следует активизировать мышью соответствующую строку ввода, если еще не задавался обработчик (при этом в модуле соответствующего объекта появится заготовка метода - обработчика события, а имя этого метода появится в строке ввода), или выбрать его из списка комбинированной строки ввода, если обработчик уже задавался, либо просто набрать его название в строке ввода. В любом случае при активизации обработчика события происходит переход к странице текстового редактора, в которой находится текст этого обработчика.

Пример

Задача. Разработать приложение, которое при нажатии на кнопку выдает фамилию и имя разработчика программы.

Ход выполнения задачи:

1 Открыть ИСР Borland C++ Builder , создать новое приложение, выбрав пункты меню: File/New Application, на экране появится форма, прдеставленная на рисунке 1.

2 Со страницы Standard библиотеки компонент перенести компонент TButton (кнопка) на форму.

3 Так же со страницы Standard библиотеки компонент перенести компонент Label (метка), для отображения фамилии и имени на форме.

4. Выделите на форме компонент Button1 - кнопку. Перейдите в Инспектор Объектов и измените ее свойство Caption (надпись), которое по умолчанию равно Button1, на "Пуск".

5. Укажите метке Label1, что надписи на ней надо делать жирным шрифтом. Для этого выделите метку, в окне Инспектора Объектов раскройте двойным щелчком свойство Font (шрифт), затем так же двойным щелчком раскройте подсвойство Style (стиль) и установите в true свойство fsBold (жирный).

6. Удалите текст в свойстве Caption метки Label1, чтобы он не высвечивался, пока пользователь не нажмет кнопку приложения. Теперь нужно написать оператор, который заносил бы в свойство Caption метки Label1 нужный текст в нужный момент, определяемый щелчком пользователя на кнопке. При щелчке в кнопке генерируется событие OnClick, обработчик которого необходимо написать.

7. Выделите кнопку Button1 на форме, перейдите в Инспектор Объектов, откройте в нем страницу событий (Events), найдите событие OnClick и сделайте двойной щелчок в окне справа от имени этого события. Вы окажетесь в окне Редактора Кода и увидите там текст:

```
void __fastcall TForm1::Button1Click(TObject *Sender)
```

{

}

Заголовок этой процедуры складывается из имени класса вашей формы (TForm1), имени компонента (Button1) и имени события без префикса On (Click).

8. Напишите в обработчике оператор задания надписи метки Label1:

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
Label1-> Caption="Иванов Петр";
}
```

9 Ваше приложение готово. Необходимо откомпилировать и выполнить его. Для этого выполните команду Run/Run, или нажмите соответствующую быструю кнопку, или нажмите "горячую" клавишу F9. После компиляции перед вами появится окно вашего первого приложения. Нажав кнопку "Пуск", вы увидите указанную вами строку текста.

10 Сохраните приложение. Для этого вызовите из главного меню команду Save Project As (Сохранить проект как) или Save All (Сохранить все) или нажмите комбинацию клавиш <Shift>+<Ctrl>+<S>. На экране отобразится стандартное диалоговое окно сохранения файлов. В этом окне необходимо выбрать место расположения проекта в имеющихся каталогах или создать для этого отдельный каталог (папку). Последний вариант предпочтительнее, поскольку позволяет систематизировать все проекты и устраняет возможность наложения файлов с одинаковыми именами друг на друга. После чего следует последовательно произвести сохранение всех файлов проекта.

Среда разработки предлагает по умолчанию свои имена для сохранения файлов. Расширение каждого сохраняемого файла указывает на его тип. Применяются следующие имена и типы файлов:

- Unit1.cpp, Unit2.cpp и т. д. – файлы с тексами программ;
- Project1.bpr, Project2.bpr и т.д. – файлы проекта;
- Project1.exe - исполняемый файл;
- *. Hpp, *.h – заголовочные файлы;

- *. Dfm - файл установок формы проекта

- файл ресурсов с расширением res

- файл проекта с расширением bpr, bpg или bpk.

Файлы с расширениями ~bpr, ~dfm,~cpp, ~h, obj, tds и exe создаются в момент компиляции проекта и могут быть восстановлены после удаления в любое время. Наибольшим объемом обладает файл с расширением tds, предназначенный для отладки программы.

Задание для самостоятельной работы

1 Запустить ИСР Visual C++ Builder. Создать свое первое приложение, изменяя число кнопок и меток и надписи на них, следуя шагам 1-10 инструкции. Задать имя формы как фамилию и имя выполняющего лабораторную работу.

2 Сохранить созданное приложение в отдельной папке.

3 Закрыть ИСР. Открыть ИСР и запустить созданное приложение.

4 Изучить все пункты меню.

Контрольные вопросы

1 Опишите возможности и преимущества ИСР Visual Builder C++? Какие еще программные продукты существуют для объектно-ориентированного программирования на языке C++?

2 Перечислите и опишите основные составляющие главного окна программы Visual Builder C++.

3 Какие функциональные вкладки представлены в библиотеке VCL?

4 Перечислите имена файлов (по умолчанию) и их расширение, создаваемые при сохранении приложения.

5 Расскажите алгоритм создания, сохранения и запуска программ в Visual Builder C++.

2 Лабораторная работа №2. Использование классов в С. Конструкторы и деструкторы

Цель работы: Реализация классов и объектов. Основные принципы ООП. Использование конструкторов и деструкторов. Работа с операциями new и delete.

Краткие теоретические сведения

Базовой идеей объектно-ориентированного подхода является объединение данных и действий, которые производятся над этими данными в единую языковую конструкцию, называемую объектом. Механизм, объединяющий данные и код, реализующий их обработку, называется инкапсуляцией. Класс представляет собой тип, создаваемый пользователем. Объект или экземпляр класса является переменной некоторого типа, определенного пользователем. Таким образом, класс это объектно-ориентированный инструмент для создания новых типов данных, являющихся объектами. Синтаксис описания класса:

```
Class имя_класса { данные-члены_класса;
                   функции-члены_класса;
} список_объектов;
```

Объявление класса только определяет тип объекта, но не создает ни одного объекта. Для создания объекта используется имя класса как спецификатор типа данных:

```
Class AnyClass {int data;
                public;
                void set_data(int, num);
                int get)data();
};
```

AnyClass ob1, ob2;

Инкапсуляция данных есть логическое связывание данных с конкретной операцией. Инкапсуляция данных означает, что данные являются не глобальными - доступными всей программе, а локальными - доступными только малой ее части. Инкапсуляция автоматически подразумевает защиту данных. Для этого в структуре class используется спецификатор раздела private, содержащий данные и методы, доступные только для самого класса. Если данные и методы содержатся в разделе public, они доступны извне класса. Раздел protected содержит данные и методы, доступные из класса и любого его производного класса.

Конструктор это функция-элемент класса, автоматически выполняющаяся в момент создания объекта. Пример использования конструктора без параметров:

```
Class MyCls { int a;
                MyCls() {a = 0}
                Void Fn1() {Тело функции Fn1();}
                Void Fn2() {Тело функции Fn2();}
};
```

Пример использования конструктора для задания даты:

```
class date {
    int month, day, year;
  public:
  date(int d =0, int m =0, int y =0);
};
date::date(int d, int m, int y)
  {
. . .
  }
```

Конструктор может быть с параметрами:

```
date today = date(23,6,1983);
```

Одновременно можно задать несколько конструкторов в программе:

```
date(int, int, int);    // день месяц год
date(char*);            // дата в строковом представлении
date(int);              // день, месяц и год сегодняшние
date();                 // дата по умолчанию: сегодня
```

Объект класса без конструкторов можно инициализировать путем присваивания ему другого объекта этого класса. Это можно делать и тогда, когда конструкторы описаны. Например:

```
date  d = today;    // инициализация посредством присваивания
```

Деструктор класса вызывается при уничтожении объекта, сам объект при этом не уничтожается. Пример вызова деструктора:

```
-MyCls() { cout <<”Destructor”;}
};
```

Для выделения памяти и ее освобождения применяют операторы new и delete. Например:

```
~date()  {
   delete day;
   delete month;
   delete year;
  }
```

Конструктор копирования является одним из важных конструкторов. Он вызывается всякий раз, когда создается новый объект, и инициализируется существующим объектом того же типа. Он имеет единственный параметр - ссылку на объект-источник. Например:

```
AnyClass ob2 = ob1; // ob1 явно инициализируется  ob2
AnyFn1(ob1); // ob1передается в качестве параметра
Ob2 = AnyFn2(); // ob2 получает значение возвращаемого объекта
```

Конструктор копирования обязателен, если в программе используются функции-элементы и переопределенные операции, которые получают формальные параметры и возвращают в качестве результата такой объект не по ссылке, а по значению.

Задание для самостоятельной работы

1. Порядок выполнения работы

1.1. Создать программу с использованием классов, определив спецификацию доступа при описании класса.

1.2. Создать конструкторы: перегружаемый, конструктор по умолчанию, конструктор копирования.

1.3. Создать деструктор.

1.4. Добавить в демонстрационную программу инициализацию объектов с помощью различных конструкторов.

2. Варианты заданий

Вариант 1.

Разработать приложение в среде C++Builder, позволяющее работать со структурой Дата: вывод текущей даты, ввод любой даты, уменьшение, увеличение на день, месяц.

Вариант 2.

Разработать приложение в среде C++Builder, позволяющее работать со структурой Время: ввод, вывод текущего времени, увеличение, уменьшение на 1 час, минуту, секунду.

Вариант 3.

Разработать приложение в среде C++Builder, позволяющее работать со структурой строки: массив для хранения строки, его длину, время создания строки. Необходимо обеспечить: изменение строки, вывод строки, нахождение подстроки в строке.

Вариант 4.

Разработать приложение в среде C++Builder, позволяющее работать со структурой окна. Необходимо обеспечить ввод следующих значений: размер окна, его положение на экране, цвет.

Вариант 5.

Разработать приложение в среде C++Builder, позволяющее работать со структурой многочлены. Структура должна включать такие поля как порядок, набор коэффициентов. Необходимо обеспечить вычисление значения многочлена для заданного параметра, вывод многочлена в удобной форме.

Вариант 6.

Разработать приложение в среде C++Builder, позволяющее работать со структурой квадратная матрица. Необходимо обеспечить: ввод матрицы, транспонирование матрицы, вывод матрицы в удобной форме.

Вариант 7.

Разработать приложение в среде C++Builder, позволяющее работать со структурой - правильные дроби. Необходимо обеспечить ввод таких значений как числитель, знаменатель и реализовать простейшие функции для работы с данными структурами: сложение, вычитание, умножение, деление.

Вариант 8.

Разработать приложение в среде C++Builder, позволяющее работать со структурой - комплексные числа. Необходимо обеспечить ввод таких значений как вещественную и мнимую часть числа, реализовать простейшие функции для работы с данными структурами: сложение, вычитание, умножение, деление.

Контрольные вопросы

1 Дайте определения понятиям: класс, объект, конструктор, деструктор

2 Как называется механизм, объединяющий данные и программный код?

3 Перечислите разделы, доступные при описании класса, и их особенности.

4 Чем отличается конструктор копирования от конструктора присваивания?

5 Для чего используется указатель this?

6 Как осуществляется доступ к элементам класса?

3 Лабораторная работа №3. Функции и объекты. Перегрузка операций

Цель работы: Освоить приемы работы с объектами, функциями, переопределенными операциями.

Краткие теоретические сведения

С классом всегда связана перегрузка операторов, при которой оператор приобретает новые свойства. Для перегрузки оператора задается оператор-функция, синтаксис которой можно описать следующим образом:

Тип имя_класса:: operator #(список_аргументов)

{выполняемые действия

}

Вместо знака # ставится знак перегружаемого оператора (+, -, *, /, %, ^, &, |, ~, ! , =, < , >, ++, --, +=, -=, *= ,/=, &=, |=, <<, >>, >>= ,<<=, [] () new delete).

Оператор в С++ - это некоторое действие или функция обозначенная специальным символом. Для того что бы распространять эти действия на новые типы данных, при этом сохраняя естественный синтаксис, в С++ была введена возможность перегрузки операторов. Выделяют следующие типы перегрузки операторов: перегрузка унарных операторов, перегрузка бинарных операторов, перегрузка операторов отношения и логических операторов, перегрузка операторов индексирования, перегрузка операторов присваивания, перегрузка операторов управления памятью, использование дружественных операторов функций, перегрузка операторов вставки и извлечения, перегрузка операторов вызова функции.

Выделяют следующие ограничения на перегрузку операторов:

1 нельзя перегружать операторы: ., *, ::, ?:, sizeof;

2 нельзя менять приоритет операторов;

3 нельзя менять число операндов оператора относительно его первоначального состояния;

4 нельзя создавать новые операции, можно только перегружать уже существующие;

5 операторы-функции не могут иметь параметров, передаваемых по умолчанию.

Функции-операции могут быть одноместными (унарными) и и двуместными (бинарными), при этом число операндов, приоритет и ассоциативность операций, определенной функцией-операцией, остаются такими же как и в базовом языке. Операции, которые имеют унарный и бинарный вариант нужно перегружать отдельно.

Порядок выполнения операций определяется их приоритетом. Но если приоритет операций одинаков, порядок их выполнения определяется ассоциативностью. Ассоциативность «слева направо» означает, что из двух операций, имеющих одинаковый приоритет, первой выполняется та, что находится слева. При ассоциативности «справа налево» первой выполняется операция, находящаяся справа.

Существует два способа описания функции для переопределения операций:

- определение функции как к дружественному классу с полным списком аргументов, в случае, если первый операнд переопределяемой операции не является объектом некоторого класса или в качестве операнда выступает сам объект;

- определение функции как обычной функции элемента класса в случае, если первый операнд переопределяемой операции является объектом некоторого класса, указатель на который передается параметром this.

Для объявления дружественной функции используется ключевое слово friend, например:

```
class   A{ int     x;   // Личная часть класса
        ...
friend  class   B;   // Функции класса B дружественны A
```

Пример

Пример перегрузки бинарных операторов:

```
#include <ioslream>
 using namespace std;
 class twins { int v1, v2;
public:
 twins() {v1=0; v2= 0;}
 twins(int m, int k) { v1 = a, v2 = Ь; }
void get_tw(int &m, int &k) { a = v1; b = v2; }
twins& operator +(twins h2);
 twins& operator -(twins h2);
twins& operator =(twins h2);
};
twins twins : : operator +(twins h2);
{temp.v1=v1+h2.v1;
temp.v2=v2+h2.v1
return temp;
}
 twins& twins :: operator =(twins h2);
{ twins temp;
v1=h2.v1;
v2=h2.v2;
return *this;
}

 int main()
{ twins am1(12, 17), am2(19, 21);am3;
int f,g;
```

```
 am3 = am1+am2;
 am3.get_tw(f,g);
cout « f « ' ' «g « endl;
am3=am1-am2;
am3.get_tw(f, g);
cout « f «' « g « end1;
am3 = am1;
 am3.get_tw(f, g);
cout « f «' « g « endl;// 20 22
 am,get_tw(f, g);
cout « f «' « g « end1;
 return 0;
}
```

Задание для самостоятельной работы

Варианты заданий

Вариант 1.

Разработать приложение в среде C++Builder, позволяющее работать с классом строки. Осуществить операции над строками: >> запись символов строки в обратном порядке; ++ нахождение наименьшего слова в строке.

Вариант 2.

Разработать приложение в среде C++Builder, позволяющее работать с классом матрица. Предусмотреть заполнение матрицы случайными числами. Осуществить операции над матрицами: ++ нахождение наибольшего значения матрицы; -- сортировка элементов матрицы по убыванию.

Вариант 3.

Разработать приложение в среде C++Builder, позволяющее работать с классом матрица. Предусмотреть заполнение двух матриц случайными числами. Осуществить операции над матрицами: + получение новой матрицы путем сложения

элементов двух заданных матриц; -- нахождение наименьшего значения полученной матрицы.

Вариант 4.

Разработать приложение в среде C++Builder, позволяющее работать с классом вектор. Предусмотреть ввод размера вектора и заполнение его случайными числами. Осуществить операции над вектором: & формирование нового вектора по условию: m[i] = (n[i]>g[i])?m[i]:g[i]; ++ найти максимальное значение элемент вектора.

Вариант 5.

Разработать приложение в среде C++Builder, позволяющее работать с классом вектор. Предусмотреть ввод размера вектора и заполнение его случайными числами. Осуществить операции над вектором: [] по заданному номеру нахождение значения элемента вектора; -- сортировка элементов вектора по возрастанию.

Вариант 6.

Разработать приложение в среде C++Builder, позволяющее работать с классом список элементов. Предусмотреть ввод длины списка и формирование второго списка путем копирования первого. Осуществить операции над списком: вставка нового элемента в список на заданное место.

Вариант 7.

Разработать приложение в среде C++Builder, позволяющее работать с классом список элементов. Предусмотреть ввод длины списка. Осуществить операции над списком: ++ сортировка элементов списка по возрастанию; -- расположение элементов списка в обратном порядке.

Вариант 8.

Создать программу с использованием классов, определив спецификацию доступа при описании класса, включить в нее функции для осуществления операций сложения: 1 переменной базового типа с объектом; 2 двух объектов класса и вычитания из объекта класса переменной базового типа.

Контрольные вопросы

1 Понятие перегрузки операций, синтаксис данной операции.

2 Перечислите ограничения на перегрузку операций.

3 Перечислите виды операций перегрузки.

4 Что такое дружественная функция?

5 Как реализовывается дружественная функция?

4 Лабораторная работа №4. Обработка ошибок

Цель работы: овладеть приемами обработки таких исключительных ситуаций как математическая ошибка (например, деление на ноль), недостаток памяти, прерывание операционной системы.

Краткие теоретические сведения

Исключительные ситуации (исключения) – это ошибки, которые возникают в процессе работы программы и вызывают ее прерывание. Для отлова и обработки таких ситуаций существует специальный оператор обработки исключений. Синтаксис оператора обработки исключений:

```
Try //пытаться
{
…
};
Catch (type1 arg ) // обработка исключения типа type1
{….
};
```

В блоке try располагаются операторы программы, во время выполнения которых необходимо обеспечить обработку исключительных ситуаций.

Блок catch осуществляет перехват и обработку исключений. При возникновении исключительной ситуации, для которой нет соответствующего оператора catch, происходит аварийное завершение программы.

Выделяют следующие виды исключений: нехватка памяти, деление на ноль, выход индекса за границы массива, арифметическое переполнение, недопустимые параметры функции (передача функции StrToInt пустую строку).

Существует также оператор throw, который используется для создания собственных исключений. Тип выражения, указанного в операторе throw,

определяет тип исключительной ситуации, а значение может быть передано обработчику прерываний.

Пример

Фрагмент программы перехвата системного исключения "деление на ноль":

```
int x = 0;
try {
std::cout <<2/x; //возникновение исключения
// Последующие операторы выполняться не будут
}
catch (...) {
std::cout << "Division by zero" << std::endl;
```

Задание для самостоятельной работы

Вариант 1.

Разработать приложение в среде C++Builder, позволяющее возводить нецелое число в отрицательную степень, предусмотреть обработку исключения.

Вариант 2.

Разработать приложение в среде C++Builder, позволяющее вычислять логарифм числа, предусмотреть обработку исключения (число отрицательное или равно нулю).

Вариант 3.

Разработать приложение в среде C++Builder, позволяющее вычислять корень числа, предусмотреть обработку исключения (число отрицательное или равно нулю).

Вариант 4.

Разработать приложение в среде C++Builder, позволяющее работать с классом вектор. Предусмотреть ввод размера вектора и заполнение его случайными числами, предусмотреть обработку исключения (выход за пределы диапазона допустимых значений).

Контрольные вопросы

1 Дайте определение исключительной ситуации.

2 Как осуществляется перехват и обработка исключительных ситуаций?

3 Примеры исключений.

5 Лабораторная работа №5. Стандартная библиотека шаблонов STL

Цель работы: освоить приемы работы с элементами библиотеки STL, такими как контейнеры, итераторы, алгоритмы.

Краткие теоретические сведения

Стандартная библиотека шаблонов STL (Standard Template Library, STL) включает четыре основных элемента: контейнеры, итераторы, алгоритмы, функциональные объекты.

Контейнеры предназначены для хранения других объектов. Различают такие типы контейнеров как массивы (vector), очереди (queue, priority_queue – очередь с приоритетом), списки (deque – двунаправленный список, list- линейный список, map – ассоциативный список для хранения пар ключ/значение), стек (stack), множество (set). Существуют методы общие для всех контейнеров:

- size() возвращает количество элементов;

- empty() – если контейнер пуст возвращает значение true;

- max_size() – максимально допустимый размер контейнера;

- begin() – возвращает итератор на начало контейнера;

-end()-возвращает итератор в конец контейнера;

- rbegin() – аналогично методу begin, но осуществление итерации в обратном порядке;

- rend()-аналогично методу end, но осуществление итерации в обратном порядке.

Помимо перечисленных методов для векторов применим методы: insert() – вставка элементов в середину вектора, erase() – удаление элементов, push_back – вставка элемента в конец вектора. Ко всем перечисленным методам к спискам могут быть применимы еще методы: merge() – слияние двух списков, pop_back – удаление последнего элемента, pop_front – удаление первого элемента.

Итераторы предоставляют методы доступа к контейнерам. Каждый итератор имеет один конструктор, функцию Restart – устанавливает итератор на первый объект, перегруженные операторы int и ++.

Алгоритмы предоставляют операции для обработки контейнеров. Для использования алгоритмов в программе необходимо подключить заголовочный файл <algorithm> . К основным алгоритмам относятся следующие алгоритмы:

- Find – поиск первого элемента по заданному значению;

- Count – определение количества элементов по заданному значению;

- Equal – сравнивает два контейнера, возвращает true если все элементы эквиваленты;
- Search – поиск последовательности значений в контейнере;
- Copy – копирование значений одного контейнера в другой контейнер;
- Swap – обменивание значений;
- Sort – сортировка значений в указанном порядке;
- Accumulate – определение суммы значений элементов в заданном диапазоне;
- For_each - выполнение указанных действий для каждого элемента контейнера.

Пример

Листинг программного кода применения алгоритмов STL к целочисленному массиву:

```
#include<iostream>
#include <algorithm>
Using namespace std;
Int main()
{ int big_arr[] = {1, 3, 5, 8, 2, 0, 7, 9, 4, 6};
Int small_arr[] = {7, 6, 2, 4};
Int i;
Int * aPtr;
For (i=0; i< 10; i++) cout <<big_arr[i] << ‘’;
Cout << endl;
aPtr = search(big_arr, big_arr + 10, small_arr, small_arr + 4);
if (aPtr == big_arr + 10) cout <<”Не найдено” << endl;
else cout << aPtr;
return0;
}
```

Задание для самостоятельной работы

Вариант 1

Разработать приложение в среде C++Builder с использованием контейнера vector из библиотеки STL: задать массив из n элементов; реализовать поиск номера наибольшего отрицательного элемента; осуществить сортировку элементов массива по возрастанию; найти среднее арифметическое элементов второй половины массива.

Вариант 2

Разработать приложение в среде C++Builder с использованием контейнера vector из библиотеки STL: задать массив из n элементов; реализовать поиск номера наибольшего элемента; осуществить сортировку элементов массива по убыванию; найти среднее арифметическое элементов первой половины массива.

Вариант 3

Разработать приложение в среде C++Builder с использованием контейнера vector из библиотеки STL: задать массив из n элементов; реализовать поиск номера наименьшего отрицательного элемента; осуществить сортировку отрицательных элементов массива по возрастанию; найти среднее арифметическое положительных элементов массива.

Вариант 4

Разработать приложение в среде C++Builder с использованием контейнера vector из библиотеки STL: задать массив из n элементов; реализовать поиск номера наименьшего элемента; осуществить сортировку второй половины элементов массива по убыванию; найти количество отрицательных элементов массива; найти сумму значений положительных элементов массива.

Вариант 5

Разработать приложение в среде C++Builder с использованием контейнера vector из библиотеки STL: задать массив из n элементов; удалить из массива элементы, значения которых находятся в диапазоне [a,b]; найти среднеарифметическое элементов массива, кратных числу g; удалить из массива элементы, имеющие нулевое значение; поменять местами первый и последний элементы массива.

Вариант 6

Разработать приложение в среде C++Builder с использованием контейнера vector из библиотеки STL: задать вещественную матрицу B[n][m]; отсортировать элементы главной диагонали по возрастанию; найти максимальный элемент третей строки матрицы; поменять местами максимальный и минимальный элемент матрицы; добавить строку в матрицу, элементы которой, будут равны разности между элементами первой строки и минимальным элементом матрицы.

Вариант 7

Разработать приложение в среде C++Builder с использованием контейнера vector из библиотеки STL: задать матрицу B[n][m]; отсортировать элементы главной диагонали по убыванию; найти минимальный элемент первого столбца матрицы;

поменять местами максимальный и минимальный элемент матрицы; заменить все отрицательные элементы матрицы на значение x.

Вариант 8

Разработать приложение в среде C++Builder с использованием контейнера vector из библиотеки STL: задать B[n][m]; найти максимальное значение в матрице среди элементов, расположенных выше главной диагонали; найти максимальный элемент последней строки матрицы; поменять местами максимальный и минимальный элемент матрицы; добавить строку в матрицу, элементы которой, будут равны сумме между элементами первой строки и минимальным элементом матрицы.

Вариант 9

Разработать приложение в среде C++Builder с использованием контейнера string из библиотеки STL: задать массив слов и подстроку; удалить из массива слов вхождение подстроки; посчитать количество оставшихся слов в строке; сформировать предложение из слов, содержащих только цифры; преобразовать строку, заменив символы ab на символы dc .

Вариант 10

Разработать приложение в среде C++Builder с использованием контейнера string из библиотеки STL: задать массив слов и подстроку; сформировать предложение из слов массива, содержащее четное количество слов и имеющее в своем составе подстроку; получить подстроку, состоящую из первых двух символов слов массива.

Контрольные вопросы

1 Что такое STL?

2 Как подключить библиотеку стандартных шаблонов?

3 Какие методы являются общими для всех контейнеров?

4 Понятие алгоритмов, основные алгоритмы STL?

5 Методы, применимые для списков и векторов?

6 Лабораторная работа №6. Визуальное программирование

Цель работы: ознакомление со свойствами и методами работы с различными компонентами библиотеки VCL среды C++ BUILDER. Работа с компонентами страницы Standard, Additional, System.

Краткие теоретические сведения

Визуальное программирование предполагает создание программ с помощью графических объектов. Программисту не нужно писать программу «с нуля», а нужно лишь добавить необходимые объекты на форму, настроить их свойства в инспекторе объектов, выбрать события, которые необходимо обработать и приложение автоматически создаст заготовки процедур и функций, выбранных событий. Разработчику останется лишь дописать или изменить программный код в созданных процедурах. Библиотека VCL состоит из более, чем ста компонент, она разделена карточными вкладками на несколько функциональных групп. Вкладка Standard - стандартная, содержит наиболее часто используемые компоненты, такие как кнопка, надпись, метка, линейка, радионабор и т.д. Назначение, свойства и события компонент данной вкладки приведены в таблице 1.

Таблица 1 – Компоненты вкладки Standard

Компонент	Назначение	Основные свойства	Основные события
1	2	3	4
Label	Текстовая надпись – отображает не редактируемый текст	Align – способ выравнивания Caption – текст надписи Autosize – если значение true, то размеры компонента преобразуются по ширине и высоте текста	

Продолжение таблицы 1

1	2	3	4
Edit	Поле ввода – для короткого текста, не имеет полосы прокрутки	Text- редактируемый текст ReadOnly - если значение true, то нельзя изменить текст в редакторе MaxLength – максимальное количество символов для ввода CharCase – преобразование текста к прописным (ecUpperCase) или строчным (ecLowerCase) буквам	OnChange – при изменении текста
Memo	Редактор текстов многострочный	Lines – текст в виде массива строк ReadOnly - если значение true, то нельзя изменить текст в редакторе MaxLength – максимальное количество символов для ввода ScrollBars - видимость полос прокрутки	OnChange – при изменении текста
CheckBox	Переключатель – имеет два значения «да», «нет»	Caption – текст рядом с переключателем Checked – определяет включен ли переключатель	OnClick – устанавливает зависимость между состоянием переключателя и состоянием других компонентов
RadioButton	Взаимоисключающий переключатель – выбор одного значения из набора вариантов	Caption – текст рядом с переключателем Checked – определяет включен ли переключатель	OnClick– зависимость между состоянием переключателя и состоянием других компонентов

Продолжение таблицы 1

1	2	3	4
RadioGroup	Группа взаимоисключающих переключателей – быстрая организация группы взаимоисключающих переключателей	Columns – число колонок в группе переключателей Caption – подпись к группе переключателей Items – подписи к переключателям ItemIndex – номер выбранного элемента, начиная с нуля	
ComboBox	Раскрывающийся список – позволяет выбрать значение из большого множества альтернатив	Items – элементы списка ItemIndex – номер выбранного элемента, начиная с нуля Sorted – сортировка элементов списка в алфавитном порядке	OnChange – происходит при вводе текста или выборе значения из списка OnВыходUp – при закрытии списка значений OnSelect – при выборе значения из списка
ListBox	Список – список элементов, которые можно просматривать и выбирать, нельзя изменять	Columns – число колонок в списке Items – элементы списка MultiSelect – возможность выбора нескольких элементов Sorted – сортировка элементов списка в алфавитном порядке	
Button	Кнопка	DefaultButton - определяет, что нажатие клавиши ENTER будет эквивалентно нажатию на данную кнопку Caption – надпись на кнопке	OnClick наступает, если пользователь осуществил щелчок основной кнопкой мыши в области компонента.

Вкладка Additional - дополнительная, является дополнением к стандартной вкладке и содержит такие компоненты как таблица, изображение, геометрическая фигура и др. Назначение, свойства и события компонент данной вкладки приведены в таблице 2.

Таблица 2 – Компоненты вкладки Additional

Компонент	Назначение	Основные свойства	Основные события
MaskEdit	Редактор с шаблоном - ввод данных в строго определенном формате	EditMask – задает шаблон для ввода текста	OnChange – при изменении текста
BitBtn	Кнопка с изображением	Glyph – файл изображения	OnClick наступает при щелчке основной кнопкой мыши в области компонента. OnDblClick наступает, при двойном щелчке кнопкой мыши в области компонента
StringGrid	Таблица строк	Options - GoEditing – возможность редактирования таблицы ColCount – количество столбцов RowCont – количество строк	OnSelectCell - возникает в момент выбора ячейки
Image	Изображение	Picture – файл с изображением Transparent - прозрачность	
Shape	Геометрическая фигура	Shape – тип фигуры (stCircle - окружность; stRectangle - прямоугольник; stSquare – квадрат) Brush – множество свойств для заливки и штриховки фигуры. Pen – множество свойств для редактирования контура фигуры	

Вкладка System - системная, содержит такие компоненты, как таймеры, плееры и ряд других. Назначение, свойства и события компонент данной вкладки приведены в таблице 3.

Таблица 3 – Компоненты вкладки System

Компонент	Назначение	Основные свойства	Основные события
1	2	3	4
Timer	Невидимый на форме элемент, с помощью которого можно отслеживать интервалы времени в программе	Interval – длина интервала, в миллисекундах	Tick - Происходит, когда указанный интервал таймера истек и таймер включен
PaintBox	Рамка рисования	Canvas – множество свойств и методов для рисования: Brush – множество свойств для заливки и штриховки фигуры Pen – множество свойств для редактирования контура фигуры	Paint – возникает в начале работы программы либо когда необходимо заново нарисовать окно
MediaPlayer	Многофункциональный проигрыватель	AutoOpen - Определяет, должно ли устройство открываться автоматически сразу после загрузки TMediaPlayer Display - Задает окно или оконный элемент управления (например, Form или Panel), в котором будет происходить отображение видео данных EnabledButtons - Определяет набор командных кнопок	OnNotify - вызывается после завершения каждой команды

Продолжение таблицы 3

1	2	3	4
OleContainer	Реализует в программе механизм внедрения и связывания объектов OLE, с помощью которого можно передавать данные между различными программами в среде Windows	OleClassName – имя класса объекта AutoActivate – способ активации объекта, может принимать значения: aaManual – программным путем; aaGetFocus – при получении фокуса; aaDoubleClick – при двойном щелчке левой кнопкой мыши. State – состояние объекта. Iconic – отображение объекта в виде иконки.	

Кроме перечисленных основных свойств компонент, у многих компонент есть такие свойства как: Name – имя компонента, Font – множество характеристик шрифта, PopupMenu – контекстное меню, TabOrder –позиция компонента в последовательности табуляции, Visible – видимость компонента во время выполнения, Enabled – доступность компонента для пользователя, Hint – текст подсказки для компонента.

Отлавливать нажатия клавиш позволяют события OnKeyDown, OnKeyUp, OnKeyPress. Событие OnKeyDown вызывается при нажатии любой клавиши на клавиатуре, событие OnKeyUp – при отпускании клавиши. Событие OnKeyPress отвечает за определенный символ таблицы ASCII, возвращает различные значения для строчных и прописных букв, в отличие от событий OnKeyDown и OnKeyUp.

Пример 1

Разработать приложение в среде C++Builder, позволяющее вводить размер матрицы, элементы матрицы, осуществить сортировку матрицы по главной диагонали.

Порядок выполнения работы:

1 Создать новое приложение

2 Добавить на форму следующие компоненты:

Edit – для задания числа строк и столбцов матрицы;

StringGrid – для хранения матрицы;

Label – различные надписи;

Button – кнопки для задания размера матрицы, осуществления сортировки, выхода из программы.

3 Настроить свойства добавленных компонент в Инспекторе объектов, согласно таблице 4.

Таблица 4 - Компоненты и описание их свойств

Название компоненты	Свойство	Значение
1	2	3
Button1	Caption	Выход
	Height	49
	Width	137
Button2	Caption	Задать размеры матрицы
	Height	33
	Width	169
Button3	Caption	Сортировка
	Height	33
	Width	169
Button4	Caption	Очистить матрицы
	Height	49
	Width	137
Edit1	Text	2
	Height	21
	Width	73
Edit2	Text	2
	Height	21
	Width	73
Lable1	Caption	Строк матрицы
	Height	13
	Width	78
Lable2	Caption	Столбцов матрицы
	Height	13
	Width	78

Продолжение таблицы 4

1	2	3
Lable3	Caption	Введите количество
	Height	13
	Width	78
Lable4	Caption	Кол-во строк = кол-во столбцов
	Height	13
	Width	164
Lable5	Caption	2 <= (кол-во)<= 10
	Height	13
	Width	164
Lable6	Caption	Неверный ввод
	Height	13
	Width	164
	Font->Color	clRed
	Visible	false
Lable7	Caption	Сортировка выполнена
	Height	13
	Width	164
	Visible	false
StringGird	ColCount	2
	DefaultColWidht	40
	DefaultRowHeight	30
	FixedCols	0
	FixedRows	0
	Height	337
	RowCount	2
	Width	433
	Option->goEditing	true

4 Осуществить обработку события Onclick для кнопок, в результате должен быть написан следующий программный код:

```
#include<vcl.h>
#pragma hdrstop
#include "Unit1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
```

```
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
int n;
void __fastcall TForm1::Button2Click(TObject *Sender) // задание размера
матрицы
{
int x, y, i, j;
{
for (i=0;i<StringGrid1->RowCount;i++)
for (j=0;j<StringGrid1->RowCount;j++)
     StringGrid1->Cells[j][i]=' ';
}
x=StrToInt(Edit1->Text);
y=StrToInt(Edit2->Text);
if ((x==y) && (x>=2) && (x<=10))
{
Label6->Visible=false;
StringGrid1->ColCount=x;
StringGrid1->RowCount=y;
}
else
Label6->Visible=true;
Label7->Visible=false;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender) // выход из программы
```

```
{
Form1->Выход();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Edit1KeyPress(TObject *Sender, char &Key)
{
if((Key!=VK_BACK)&&(Key<'0'||Key>'9'))Key=2; //ne raven udaleniu
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Edit2KeyPress(TObject *Sender, char &Key)
//vvodtolkocifr
{
if((Key!=VK_BACK)&&(Key<'0'||Key>'9'))Key=2;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button4Click(TObject *Sender) // очищение матрицы
{
inti, j;
for (i=0;i<StringGrid1->RowCount;i++)
for (j=0;j<StringGrid1->RowCount;j++)
    StringGrid1->Cells[i][j]=' ';
 Label7->Visible=false;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button3Click(TObject *Sender) // сортировка матрицы
{
inti, j, b, a[11];
for (i=0;i<(StringGrid1->RowCount);i++){
a[i]=StrToInt(StringGrid1->Cells[i][i]);
 }
```

```
for (j=0;j<(StringGrid1->RowCount-1);j++){
for (i=0;i<(StringGrid1->RowCount-1);i++)
if (a[i]<=a[i+1]){}
else {b=a[i];
a[i]=a[i+1];
a[i+1]=b;
 }
 }
for (i=0;i<(StringGrid1->RowCount);i++){
 StringGrid1->Cells[i][i]=IntToStr(a[i]);
 }
 Label7->Visible=true;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::StringGrid1KeyPress(TObject *Sender, char &Key)
{
if((Key!=VK_BACK)&&(Key<'0'||Key>'9'))Key=0;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::StringGrid1Click(TObject *Sender)
{
 Label7->Visible=false;
}
```

5 Входная и результирующая экранные формы работы приложения представлена на рисунке 2 и рисунке 3.

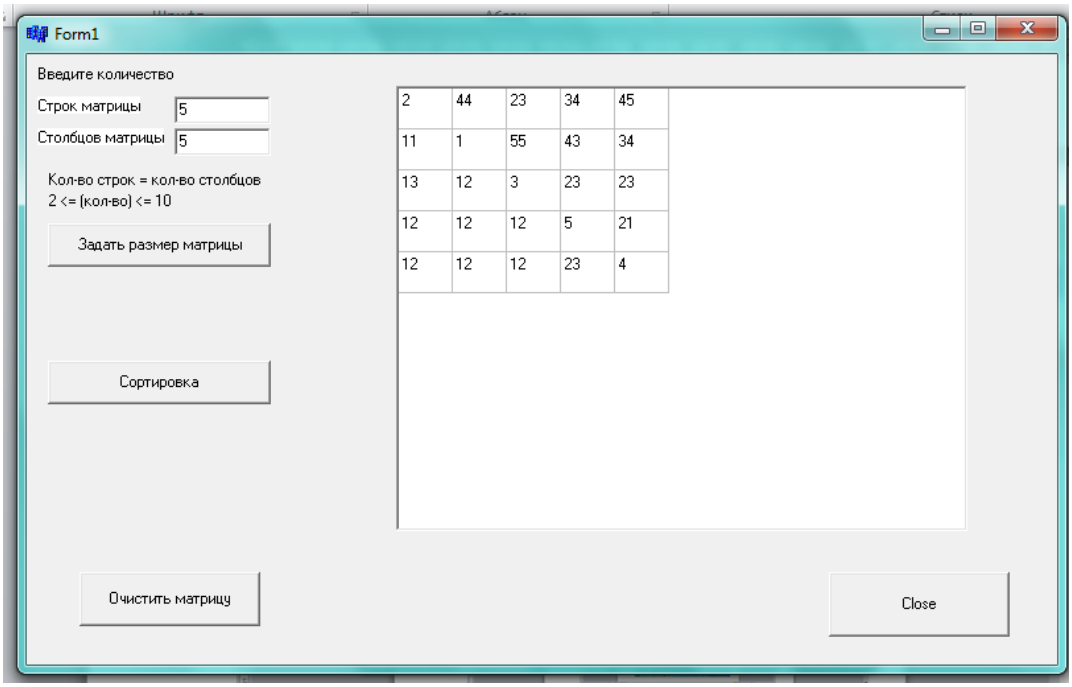

Рисунок 2 – Входная экранная форма программы

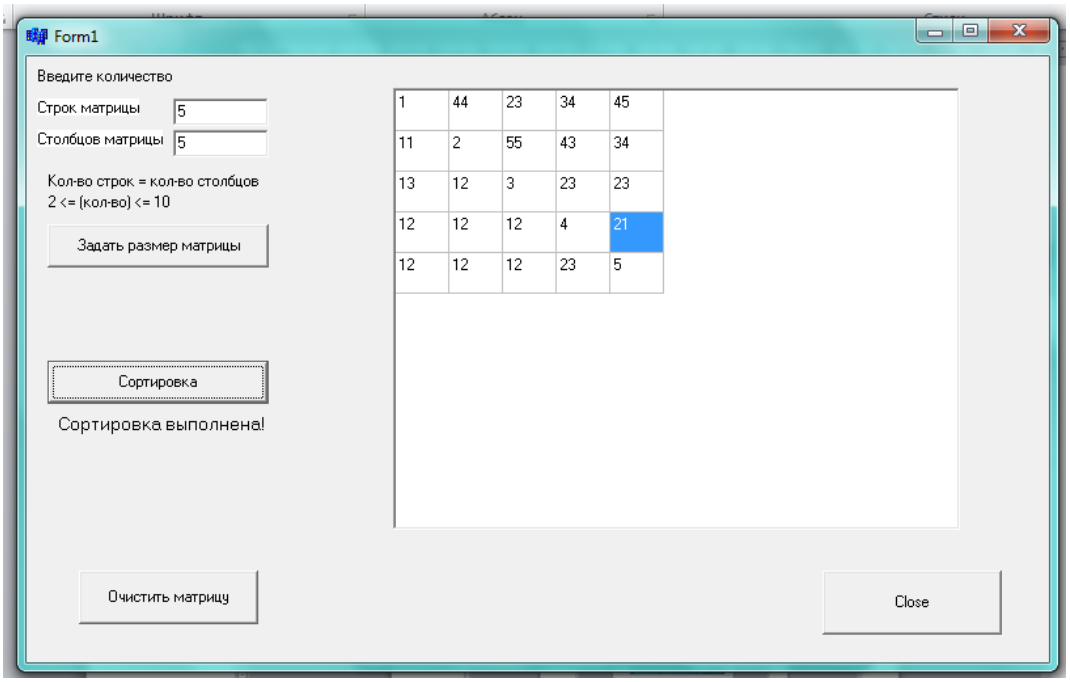

Рисунок 3 – Результирующая экранная форма работы программы.

Пример 2

Разработать приложение в среде C++Builder, позволяющее осуществлять задание размера таблицы с помощью полосы прокрутки в диапазоне 2X2, 7X7 и отображать текущий размер таблицы. По умолчанию размер таблицы 4X4.

Порядок выполнения работы:

1 Создать новое приложение

2 Добавить на форму следующие компоненты:

Edit – для отображения размерности матрицы;

StringGrid – таблица;

ScrollBar – полоса прокрутки;

Label – различные надписи;

Button – кнопка для выхода из программы.

3 Настроить свойства добавленных компонент в Инспекторе объектов, согласно таблице 5.

Таблица 5 - Компоненты и описание их свойств

Название компоненты	Свойство	Значение
1	2	3
Button1	Caption	Выход
	Height	57
	Width	137
Edit1	Text	4x4
	Height	24
	Width	105
Lable1	Caption	Текущий размер таблицы
	Height	16
	Width	171
ScrollBar1	Height	17
	Width	249
	Kind	sbHorizontal
	Max	7
	Min	2
ScrollBar2	Height	249
	Kind	sbVertical
	Max	7
	Min	2

Продолжение таблицы 5

1	2	3
StringGird	ColCount	4
	DefaultColWidht	34
	DefaultRowHeight	34
	FixedCols	0
	FixedRows	0
	Height	249
	RowCount	4
	Width	249

4 Осуществить обработку события OnScroll, в результате должен быть написан следующий программный код:

```
#include<vcl.h>
#pragmahdrstop
#include "Unit1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm1::ScrollBar1Scroll(TObject *Sender,
TScrollCodeScrollCode, int&ScrollPos)
{
 StringGrid1->ColCount=ScrollPos;
 Edit1->Text=IntToStr(StringGrid1->ColCount)+'x'+IntToStr(StringGrid1->RowCount);
}
//---------------------------------------------------------------------------
```

```
void __fastcall TForm1::ScrollBar2Scroll(TObject *Sender,
TScrollCodeScrollCode, int&ScrollPos)
{
StringGrid1->RowCount=ScrollPos;
Edit1->Text=IntToStr(StringGrid1->ColCount)+'x'+IntToStr(StringGrid1->RowCount);
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
Form1->Выход();
}
```

5 Входная и результирующая экранные формы работы приложения представлены на рисунке 4 и рисунке 5.

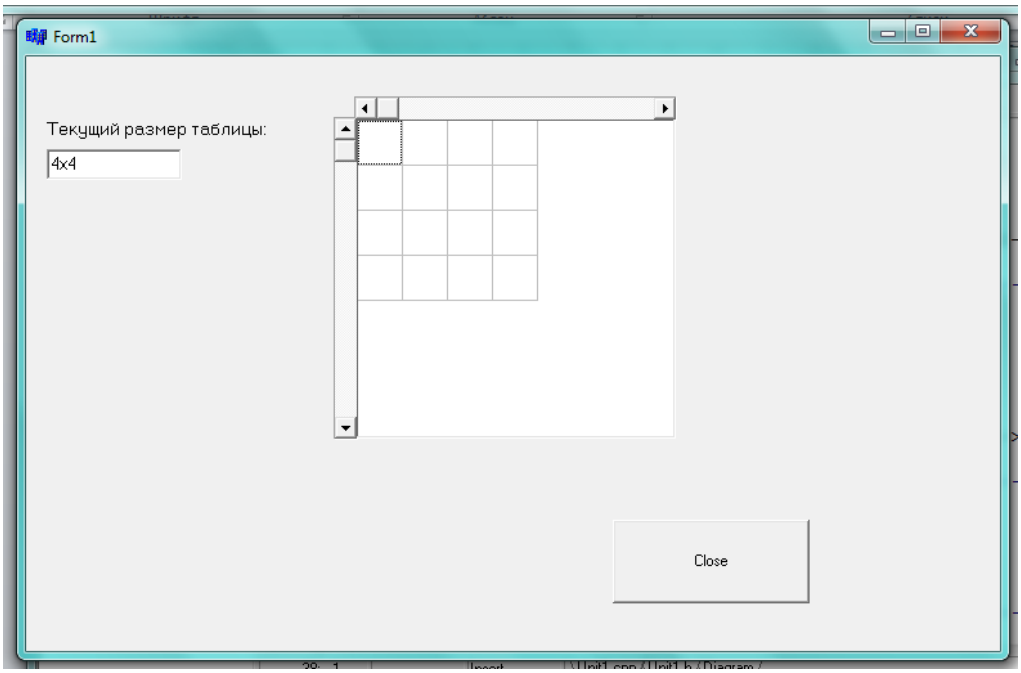

Рисунок 4 – Входная экранная форма приложения

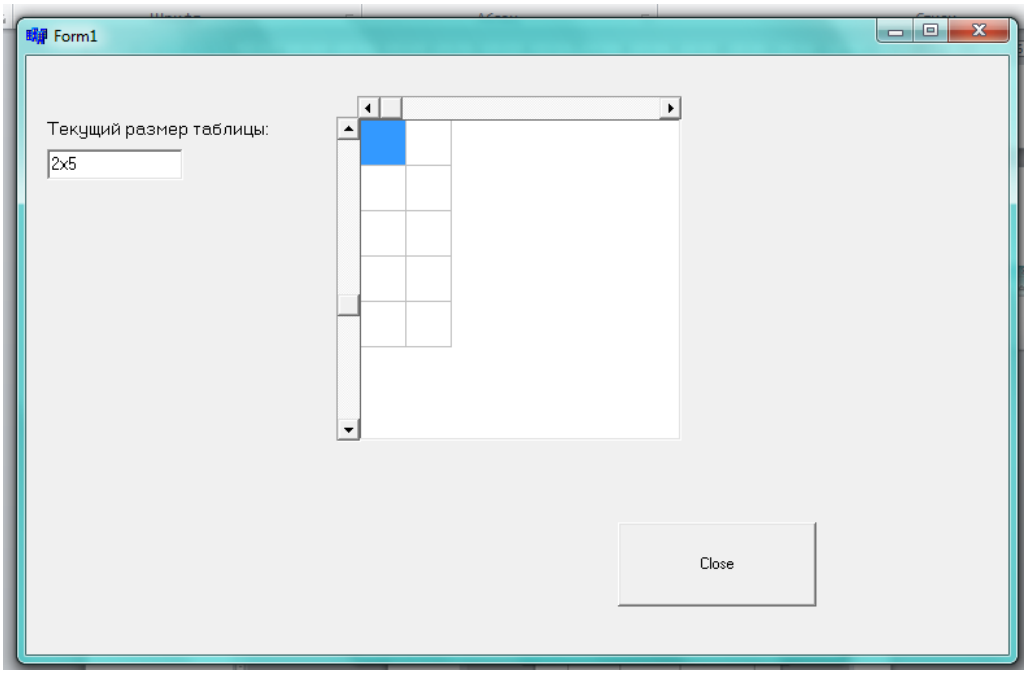

Рисунок 5 – Результирующая экранная форма работы приложения

Пример 3

Разработать приложение в среде C++Builder, отображающее календарь за декабрь текущего года с горизонтальным расположением недель и позволяющее осуществлять движение рамки (цвет - синий, форма - окружность) по датам календаря (дискрет времени 0.6 сек). Запуск движения – команда контекстного меню, остановка – кнопка «Стоп», что приводит к установке рамки на первую дату.

Порядок выполнения работы:

1 Создать новое приложение

2 Добавить на форму следующие компоненты:

Image – для отображения календаря;

Shape – рамка для движения по датам;

Timer – для организации движения рамки;

PopupMenu– контекстное меню;

Button – кнопка для выхода из программы.

3 Настроить свойства добавленных компонент в Инспекторе объектов, согласно таблице 6.

Таблица 6 – Компоненты и описание их свойств

Объект	Свойство	Значение
Image1	Picture	(TJPEGImage)
Image1	Stretch	true
Button1	Caption	Стоп
Shape1	Brush.Style	bsClear
Shape1	Shape	stCircle
Shape1	Pen.Width	1
Shape1	Pen.Color	clBlue
Timer1	Enabled	False
Timer1	Interval	600
PopupMenu1	Items	Пуск
Form1	PopumMenu	PopumMenu1

4 Осуществить обработку события OnPopup, в результате должен быть написан следующий программный код:

```
#include<vcl.h>
#pragmahdrstop
#include "Unit1.h"
//---------------------------------------------------------------------------
#pragmapackage(smart_init)
#pragmaresource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
      : TForm(Owner)
{
}

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

void __fastcall TForm1::Timer1Timer(TObject *Sender)
```

```
{
Shape1 -> Left = Shape1 -> Left + 68;
if(Shape1 -> Left == 532)
{
Shape1 -> Left = 56;
Shape1 -> Top = Shape1 -> Top + 44;
}
if(Shape1 -> Left == 260 && Shape1 -> Top == 284)
Timer1 ->Enabled = false;
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button2Click(TObject *Sender)
{
Timer1 -> Enabled = false;
Shape1 -> Left = 124;
Shape1 -> Top = 108;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::N1Click(TObject *Sender)
{
Timer1->Enabled = true;
}
        //---------------------------------------------------------------------------
```

5 Результирующая экранная форма работы приложения представлена на рисунке 6.

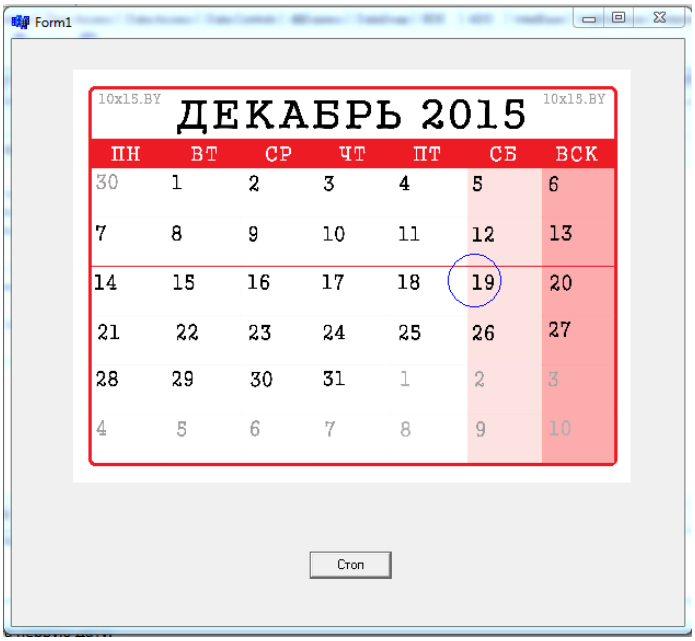

Рисунок 6 – Результирующая экранная форма работы приложения

Задание для самостоятельной работы

1 Порядок выполнения работы

1.1 Разработать приложение в среде C++Builder согласно варианту.

1.2 Оформить отчет по лабораторной работе согласно плану, приведенному в примерах.

2 Варианты заданий для работы с вкладкой Standard

1. Отобразить следующие компоненты: метка Label с указанием фамилии студента; ListBox с опциями-цифрами 1, 2, 3, 4; кнопки вверх, вниз, вправо, влево, закрыть; панель Panel; CheckBox – отображение видимости. При нажатии мышью кнопки метка с фамилией смещается в соответствующем направлении на количество пикселей, выбранных в ListBox. Текущий шаг перемещения отображается на панели, если она видна. Режим видимости последней задается флагом CheckBox.

2. Отобразить следующие компоненты: метка Label с указанием номера группы; ListBox с опциями вверх, вниз, вправо, влево; радиогруппа для выбора шага перемещения на 1, 2 или 3 пикселя; ScrollBar, для определения размера шрифта на метке в диапазоне 8-16 пикселей; кнопка «Сдвиг» и кнопка «Закрыть». При выборе

мышью соответствующей опции с направлением и нажатии кнопки «Сдвиг» форма смещается в соответствующем направлении на 1 пиксель

3. Отобразить следующие компоненты: четыре кнопки с номерами, редактор Edit, панель, флажок CheckBox, кнопки “Очистить” и «Закрыть». Исходно окно редактора пусто. Нажатие кнопки с номером приводит к появлению в окне редактора или на панели ее номера, добавляемого в конец строки. Переключение вывода производится флажком CheckBox. Кнопка «Очистить» приводит окно редактора или поверхность панели в исходное состояние.

4. Отобразить следующие компоненты: метка Label, четыре кнопки с номерами, ScrollBar с разметкой от 1 до 4 и кнопка «Закрыть». При нажатии мышью какой-либо кнопки она становится невидимой, но появляется ранее скрытая кнопка. Номер невидимой кнопки появляется на метке. Синхронно перемещается ползунок линейки ScrollBar. Перемещение ползунка мышью приводит к вышеописанным манипуляциям с кнопками.

5. Отобразить следующие компоненты: редактор Edit (изначально отображается число 20), кнопки «плюс» и «минус», список ListBox с опциями "Показать" и "Скрыть", кнопка «Очистить» и кнопка Выход Нажатие кнопки «плюс» приводит к увеличению содержимого редактора на 1, а кнопки «минус», к уменьшению на 1. При выборе опции "Скрыть" кнопки исчезают, а опции "Показать" появляются. Нажатие кнопки «Очистить» приводит к установке редактора в исходное состояние. Выход заканчивает программу.

6. Отобразить следующие компоненты: редактор Memo, в центре окна; четыре кнопки вверх, вниз, вправо, влево; список ComboBox с опциями 1, 2, 3 и кнопка Выход. Нажатие кнопки приводит к перемещению правой или нижней границы окна редактора в соответствующем направлении на количество пикселей, заданных в компоненте ComboBox. При этом в окне редактора отображается его текущий размер. Если редактор уменьшается до размера текста, то кнопка уменьшения для соответствующего направлении исчезает. При увеличении данного размера она снова появляется. Кнопка Выход заканчивает программу.

7. Отобразить следующие компоненты:. редактор Memo, занимающий центр окна; компонент ListBox с четырьмя опциями вверх, вниз, вправо, влево; радионабор с вариантами "Левая и верхняя" и "Правая и нижняя"; кнопка «Пуск». Нажатие последней приводит к перемещению на 1 пиксель в соответствии с опцией, указанной в компоненте ListBox, границ окна, заданных радионабором. Выбор дублируется в редакторе Memo. Если опция не выбрана, то в окне редактора выводится соответствующее сообщение.

8. На форме располагаются: редактор Edit, занимающий центр окна; компонент ListBox с четырьмя опциями вверх, вниз, вправо, влево; линейка ScrollBar, размеченная пятью цветами и кнопка «Пуск». Нажатие этой кнопки приводит к отображению в окне редактора слов «Влево», «Вправо», «Вверх», «Вниз» в соответствии с опцией, указанной в компоненте ListBox. Цвет надписи задается компонентом ScrollBar. Если опция не выбрана, то в окне редактора выводится соответствующее сообщение.

9. Отобразить следующие компоненты: четыре метки Label с названиями цветов, список ListBox с опциями-цветами, RadioGroup с теми же цветами, флажок CheckBox, кнопка «Выбор» и кнопка «Закрыть». При выборе названия цветка и нажатии мышью кнопки «Выбор» или клавиши "Enter" на клавиатуре соответствующая метка исчезает, а спрятанная появляется. Вариант выбора: через список или радиогруппу, - задается флажком. Кнопка Выход заканчивает программу.

10. Отобразить следующие компоненты: метки Label с названиями городов, список ListBox с опциями-городами, линейка Scrollbar для изменения количества городов в диапазоне от 4 до 8, кнопка «Пуск» и кнопка «Закрыть». Исходно все метки-города скрыты. При выборе названия города в ListBox и нажатии мышью кнопки «Пуск» соответствующая метка появляется, а ранее видимая исчезает. Кнопка Выход заканчивает программу.

3 Варианты заданий для работы с вкладкой Additional

1. Отобразить следующие компоненты: Edit, ListBox, Label, контекстное меню для формы и две кнопки (с надписями «Очистить» и «Закрыть»). При запуске

программы курсор находится в Edit. Вводится с проверкой дробное число как аргумент тригонометрической функции. Сама функция: sin, cos, tg или ctg, - выбирается с помощью ListBox. Над окнами ввода аргумента и выбора функции надписи «Аргумент» и «Функция». Запуск на вычисление производится с помощью контекстного меню. Результат помещается на Label. Очистка окна аргумента производится кнопкой. Вторая кнопка заканчивает программу.

2. Отобразить следующие компоненты: два редактора Edit, , Label, главное меню, две кнопки (с надписями «Очистить» и «Закрыть»). При запуске программы курсор находится в одной из строк Edit. В первом окне вводится с проверкой дробное число, а во второе - целое как показатель возведения в степень первого числа. Над окнами ввода аргумента и показателя степени надписи «Аргумент» и «Показатель». Запуск на вычисление производится с помощью команды главного меню. Результат помещается на Label. Очистка окна аргумента производится кнопкой. Вторая кнопка заканчивает программу.

3. Отобразить следующие компоненты: редакторы MaskEdit и Memo, Label, ListBox, кнопка BitBtn вида Yes и кнопка Выход. При запуске программы курсор находится в редакторе MaskEdit. В него вводится строка, преобразуемая к верхнему регистру. При нажатии кнопки Yes она копируется в Memo или на Label. Приемник копирования выбирается с помощью ListBox. Кнопка Выход заканчивает программу.

4. Отобразить следующие компоненты: два компонента ListBox, таблица StringGrid размером 4x6, две кнопки Button и кнопка Выход. На одной из кнопок Button надпись «Очистка», на другой – «Занесение». Над одним из компонентов ListBox надпись «Строки», над другим – «Столбцы». С помощью первого выбирается строка таблицы, другим - столбец. При нажатии кнопки “Занесение” в соответствующую ячейку заносится произведение номеров строки и столбца. Кнопка «Очистка» очищает данную ячейку. Кнопка Выход заканчивает программу.

5. Отобразить следующие компоненты: таблица StringGrid 3x6, шесть кнопок Button с номерами 1 – 6 и кнопка Выход. Нажатие кнопки с номером

приводит к появлению 1 в соответствующей ячейке второй строки таблицы. Остальные ячейки этой строки при этом очищаются.

6. На форме располагаются: таблица StringGrid 4x5, два набора радиокнопок с соответствующими надписями для выбора номеров строк и столбцов таблицы, кнопка Button с надписью «Занесение» и кнопка Выход. Выбор ячейки таблицы и нажатие кнопки «Занесение» приводит к появлению в данной ячейке названия группы. При этом в другой ячейке надпись исчезает. Кнопка Выход заканчивает программу.

7. Отобразить следующие компоненты: таблица StringGrid 3x6, шесть кнопок Button с номерами 1 – 6 для выбора столбца, компонент ListBox для выбора строки и кнопка Выход. Нажатие кнопки с номером приводит к появлению в соответствующей ячейке таблицы суммы номеров заданных строки и столбца. Остальные ячейки таблицы при этом очищаются.

8. На форме располагаются: шесть разных кнопок BitBtn с цифрой и буквой сверху, радиокнопка для задания режима работы, кнопка Выход и редактор Edit. Нажатие каждой кнопки приводит к отображению в окне редактора цифры или буквы на кнопке. Режим отображения «цифра/буква» задается радиокнопкой. Кнопка Выход заканчивает программу.

9. На форме располагаются: шесть разных кнопок BitBtn с номерами сверху, кнопка Выход и таблица StingGrid 6x6. Нажатие кнопки приводит к последовательному сокрытию одной из кнопок и показу ранее скрытой кнопки. При этом номер ранее скрытой кнопки отображается в соответствующей ячейке на главной диагонали таблицы. Нажатие клавиши «Пробел» на клавиатуре приводит к очистке главной диагонали. Кнопка Выход заканчивает программу.

10. На форме располагаются: семь разных кнопок Button с названиями-днями недели, кнопка Выход, флажок CheckBox и редактор Edit. Нажатие каждой кнопки приводит к отображению соответствующего дня недели в редакторе Edit, а флажок задает режим отображения: заглавными или прописными буквами. Кнопка Выход заканчивает программу.

4 Варианты заданий для работы с вкладкой System

1. На пространстве формы изображен календарь за январь текущего года с горизонтальным расположением недель. Рамка красного цвета в виде квадрата с толщиной линии в 1 пиксель движется по датам с дискретом времени 0.4 сек. Запуск движения – кнопка «Пуск», остановка – кнопка «Стоп», что приводит к установке рамки на первую дату.

2. На пространстве формы изображен календарь за февраль текущего года с горизонтальным расположением недель. Рамка черного цвета в виде квадрата с закругленными углами и толщиной линии в 2 пикселя движется по датам с дискретом времени 0.5 сек. Запуск движения – контекстное меню, остановка – кнопка «Стоп», что приводит к установке рамки на первую дату.

3. На пространстве формы изображен календарь за март текущего года с горизонтальным расположением недель. Рамка синего цвета в виде окружности с толщиной линии в 3 пикселя движется по датам с дискретом времени 0.6 сек. Запуск движения – кнопка «Пуск», остановка - контекстное меню, что приводит к установке рамки на первую дату.

4. На пространстве формы изображен календарь за апрель текущего года с горизонтальным расположением недель. Рамка зеленого цвета в виде квадрата с толщиной линии в 2 пикселя движется по датам с дискретом времени 0.7 сек. Запуск движения – кнопка «Пуск», остановка – главное меню, что приводит к установке рамки на первую дату.

5. На пространстве формы изображен календарь за май текущего года с горизонтальным расположением недель. Рамка красного цвета в виде квадрата с закругленными углами и толщиной линии в 3 пикселя движется по датам с дискретом времени 0.4 сек. Запуск движения – главное меню, остановка – кнопка «Стоп», что приводит к установке рамки на первую дату.

6. На пространстве формы изображен календарь за июнь текущего года с горизонтальным расположением недель. Рамка черного цвета в виде окружности с толщиной линии в 1 пиксель движется по датам с дискретом времени 0.5 сек. Запуск движения и остановка движения – команды главного меню, что приводит к установке рамки на первую дату.

7. На пространстве формы изображен календарь за июль текущего года с горизонтальным расположением недель. Рамка зеленого цвета в виде квадрата с толщиной линии в 2 пикселя движется по датам с дискретом времени 0.6 сек. Запуск движения и остановка движения – команды контекстного меню, что приводит к установке рамки на первую дату.

8. На пространстве формы изображен календарь за май текущего года с вертикальным расположением недель. Рамка красного цвета в виде квадрата с закругленными углами и толщиной линии в 3 пикселя движется по датам с дискретом времени 0.6 сек. Запуск движения – кнопка «Пуск», остановка – команда главного меню, что приводит к установке рамки на первую дату.

9. На пространстве формы изображен календарь за июнь текущего года с вертикальным расположением недель. Рамка черного цвета в виде окружности с толщиной линии в 1 пиксель движется по датам с дискретом времени 0.7 сек. Запуск движения – команда главного меню, остановка – кнопка «Стоп», что приводит к установке рамки на первую дату.

10. На пространстве формы изображена шахматная доска. Шашка красного цвета движется по периметру доски по часовой стрелке с дискретом 0.8 сек. Запуск движения – команда главного меню, остановка – кнопка «Пуск», что приводит к установке шашки на левую верхнюю клетку.

11. На пространстве формы изображена шахматная доска. Шашка синего цвета движется по периметру доски против часовой стрелки с дискретом 0.4 сек. Запуск и остановка движения – команды главного меню, что приводит к установке шашки на правую верхнюю клетку.

12. На пространстве формы изображена шахматная доска. Шашка зеленого цвета движется по главной диагонали доски взад-вперед с дискретом 0.5 сек. Запуск движения – команда контекстного меню, остановка – кнопка «Стоп», что приводит к установке шашки на правую нижнюю клетку.

13. На пространстве формы изображена шахматная доска. Шашка желтого цвета движется по вспомогательной диагонали доски с дискретом 0.6 сек. Запуск

движения – кнопка «Пуск», остановка – команда главного меню, что приводит к установке шашки на левую нижнюю клетку.

14. На пространстве формы изображен квадрат размером 250x250. По часовой стрелке по периметру движется окружность диаметром 40 пикселей По верхней стороне движение идет со скоростью 200/сек. Далее на каждой стороне скорость уменьшается в два раза. Запуск движения – команда главного меню, остановка – кнопка «Пуск», что приводит к установке окружности в левый верхний угол квадрата.

15. На пространстве формы изображена окружность диаметром 300 пикселей. По ней против часовой стрелки движется окружность диаметром 40 пикселей со скоростью100/сек. Запуск и остановка движения – команды главного меню, что приводит к установке малой окружности в верхнюю точку большой окружности.

16. На пространстве формы изображена окружность диаметром 350 пикселей. Внутри ее, касаясь контура, по часовой стрелки движется окружность диаметром 40 пикселей со скоростью 200/сек. Окружности разного цвета и толщины. Запуск движения – кнопка «Пуск», остановка – команда главного меню, что приводит к установке малой окружности в самое верхнее положение.

17. На пространстве формы изображен прямоугольник размером 250x300. Внутри его, касаясь контура, против часовой стрелки движется квадрат 30x30 пикселей со скоростью 200/с. При смене направления меняется цвет квадрата. Запуск и остановка движения – команды главного меню, что приводит к установке квадрата в левый верхний угол.

Контрольные вопросы

1 Сходства и отличия компонентов Lable, Memo, Edit?

2 Перечислите свойства, доступные большинству компонент библиотеки VCL?

3 Назначение компонента Timer, примеры применения?

4 Перечислите события, возникающие при нажатии клавиш клавиатуры?

5 Какой компонент необходим для рисования различных геометрических фигур?

6 Какое свойство отвечает за видимость компонент на форме?

7 В каком свойстве компонент необходимо указывать текст подсказки, который будут отображаться при наведении курсора на компонент?

8 Настройка какого свойства компонента StringGrid, дает возможность редактировать текст ячейки?

9 Какой компонент необходим для настройки контекстного меню? На какой вкладке VCL его можно найти?

10 Назовите компонент, отображающий раскрывающий список?

Список использованных источников

1Ашарина И.В. Объектно-ориентированное программирование в C++: лекции и упражнения. / И.В. Ашарина. – М. Горячая линия –Телеком, 2008. – 320 с. ISBN 978-5-9912-0038-7

2 Павловская Т.А. C++. Объектно-ориентированное программирование: практикум. / Т.А. Павловская, Ю.А. Щупак. – Спб.: Питер, 2008. – 265 с. ISBN 978-5-94723-842-6

3 Хорев, П. Б. Технологии объектно-ориентированного программирования. / П. Б. Хорев .- 2-е изд., стер. - М. : Академия, 2008. - 448 с. - (Высшее профессиональное образование). - Библиогр.: с. 445-446. - ISBN 978-5-7695-5262-5.

4 Бабушкина, И. А. Практикум по объектно-ориентированному программированию [Электронный ресурс] / И. А. Бабушкина, С. М. Окулов. - 3-е изд. (эл.). - М. : БИНОМ. Лаборатория знаний, 2012. - 366 с. : ил. ; 60x90/16. - ISBN 978-5-9963-0954-2. http://znanium.com/catalog.php?bookinfo=366434

5 Галявов, И. Р. Borland C++ для себя [Электронный ресурс] / И. Р. Галявов. - М.: ДМК Пресс, 2009. - 432 с.: ил. - (Самоучитель). - ISBN 5-94074-094-4. http://znanium.com/bookread2.php?book=408232