

Министерство образования и науки Украины
Донбасская государственная машиностроительная академия

ПРОГРАММИРОВАНИЕ И АЛГОРИТМИЧЕСКИЕ ЯЗЫКИ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к самостоятельной работе

(для студентов направления «Системный анализ» заочной формы обучения)

Часть 2

Утверждено
на заседании кафедры ИСПР
Протокол № 2 от 9 сентября 2014г.

Краматорск 2014

Программирование и алгоритмические языки : Методические указания к самостоятельной работе для студентов направления «Системный анализ» заочной формы обучения. Часть 2 / Сост. А. Ю. Мельников. – Краматорск: ДГМА, 2014. – 16 с.

Содержат методические указания по подготовке к выполнению контрольной работы и сдачи экзамена по дисциплине «Программирование и алгоритмические языки» студентами заочной формы обучения.

Составитель	Мельников А.Ю., канд. техн. наук, доцент
-------------	--

Отв. за выпуск	Мельников А.Ю., канд. техн. наук, доцент
----------------	--

СОДЕРЖАНИЕ

Общие сведения	4
Задача №1. Создание метода – реакции на событие в среде визуального программирования	4
Задача №2. Описание класса на языке программирования C++	5
Перечень вопросов к экзамену	6
Список рекомендуемой литературы	10
Приложение А. Справочные материалы	12

ОБЩИЕ СВЕДЕНИЯ

Контрольная работа предполагает выполнение следующих заданий:

1. описание метода – реакции на заданное событие в среде визуального программирования, основанной на языке Object Pascal – 40 баллов;
2. описание заданного класса, являющегося потомком других заданных классов и содержащего определенные свойства и методы, на языке программирования C++ – 60 баллов.

Контрольная работа считается сданной в случае набора не менее 40 баллов.

Экзамен представляет собой ответы на 15 теоретических вопросов, представленных в виде тестов закрытой формы (выбор одного или нескольких вариантов ответа из предложенного перечня) – 100 баллов (тема «Среда визуального программирования»: 5 вопросов по 10 баллов за каждый полностью правильный ответ; тема «Объектно-ориентированный подход»: 6 вопросов по 5 баллов за каждый полностью правильный ответ; тема «Объектно-ориентированное программирование в среде C++»: 4 вопроса по 5 баллов за каждый полностью правильный ответ).

Экзамен считается сданным в случае набора не менее 55 баллов – как по самому экзамену, так и в среднем между контрольной и экзаменационной работой.

ЗАДАЧА №1

СОЗДАНИЕ МЕТОДА – РЕАКЦИИ НА СОБЫТИЕ В СРЕДЕ ВИЗУАЛЬНОГО ПРОГРАММИРОВАНИЯ

Описать метод – реакцию на заданное событие в среде визуального программирования, основанной на языке Object Pascal.

Примеры выполнения задания

1. Создается приложение, на форме которого размещаются три компонента – кнопка и два поля ввода текста. Сделать так, чтобы по нажатию на кнопку содержание первого поля ввода текста скопировалось во второе.

```
procedure TForm1.Button1Click(Sender: TObject);  
begin  
    Edit2.Text:=Edit1.Text  
end;
```

2. Создается приложение, на форме которого размещаются три компонента – кнопка, список без возможности ввода нового значения и метка. Сделать так, чтобы по нажатию на кнопку номер выбранной темы был присвоен значению метки.

```
procedure TForm1.Button1Click(Sender: TObject);  
begin  
    Label1.Caption:= IntToStr(ListBox1.ItemIndex)  
end;
```

3. Создается приложение, на форме которого размещаются четыре компонента – кнопка, список с возможностью ввода нового значения, флажок и поле вывода многострочного текста. Сделать так, чтобы по нажатию на кнопку в случае «включенного» флажка в поле многострочного текста выводился номер выбранной темы из списка, а в случае «выключенного» флажка – сама строка, введенная в поле списка или выбранная из него.

```
procedure TForm1.Button1Click(Sender: TObject);  
var i,k: integer;  
begin  
    Memo1.Lines.Clear;  
    if CheckBox1.Checked then begin  
        k:=0;  
        for i:=1 to ComboBox1.Items.Count do  
            if ComboBox1.Text = ComboBox1.Items[i-1] then  
                k:=i;  
            if k = 0 then Memo1.Lines.Add('Совпадений не найдено!')  
            else Memo1.Lines.Add(IntToStr(k))  
        end else Memo1.Lines.Add(ComboBox1.Text)  
    end;  
end;
```

ЗАДАЧА №2

ОПИСАНИЕ КЛАССА НА ЯЗЫКЕ ПРОГРАММИРОВАНИЯ C++

Описать заданный класс, являющийся потомком других классов и содержащий заданные свойства и методы, на языке программирования C++.

Примеры выполнения задания

Описать класс «Точка», являющийся потомком классов «Графический режим» и «Графический объект», содержащий координаты точки и методы ее скрытия и отображения.

```

class point: public gr_mode, gr_obj
{
public:
    point(int xp=0, int yp=0, int col=7): gr_obj(col)
    {
        setpx(xp);
        setpy(yp);
        show();
    }
    ~point()
    {
        hide();
    }
    int getpx()
    {
        return x;
    }
    void setpx(int px)
    {
        x=px;
    }
    int getpy()
    {
        return y;
    }
    void setpy(int py)
    {
        y=py;
    }
    void show()
    {
        putpixel(x,y,getcolor());
    }
    void hide()
    {
        putpixel(x,y,getbkcolor());
    }
private:
    int x,y;
};

```

ПЕРЕЧЕНЬ ВОПРОСОВ К ЭКЗАМЕНУ

Тема 1. Среда визуального программирования

1. «Инспектор Объектов» состоит из двух страниц, которые называются...
2. Поместить главное меню в программу можно через компоненту...
3. Поместить всплывающее меню в программу можно через компоненту...
4. Простое отображение текста на экране осуществляет компонента...
5. Для отображения и ввода короткого фрагмента текста используется компонента...
6. Для ввода большого текста в программу используется компонента...
7. «Кнопку» на экране отображает компонента...
8. Выбрать один вариант из нескольких позволяет компонента...
9. Сделать выбор по принципу «да/нет» позволяет компонента...
10. Прокручиваемый список с возможностью ввода дополнительной информации в верхней части окошка позволяет компонента...
11. Прокручиваемый список без возможности ввода дополнительной информации в верхней части окошка позволяет компонента...
12. Вывести одинарную полосу прокрутки позволяет компонента...
13. Вывести две полосы прокрутки (вертикальную и горизонтальную) позволяет компонента...
14. Имена методов в Delphi (Lazarus) образуются согласно...
15. Имя цвета в Delphi (Lazarus) представляет собой его название на английском языке с добавлением...
16. С любым проектом в Delphi (Lazarus) связано файлов...
17. Главный файл Delphi (Lazarus)-проекта имеет расширение...
18. Модули Delphi (Lazarus)-программ имеют расширения...
19. Файл главной формы Delphi (Lazarus)-проекта имеет расширение...
20. Компонента TImage может принимать изображения таких графических форматов...
21. Свойство «Canvas» есть у...
22. Внешний вид и поведение формы (компонента) определяют...
23. Свойства, значения которых являются числами или строками, называются...
24. Свойства, которые могут принимать значения из предопределенного набора (списка), называются...
25. Свойства, значения которых поддерживают вложенные значения или объекты, называются...
26. Категория объектов, обладающих одинаковыми свойствами и поведением, называется...
27. Процедура, которая определена как часть класса и инкапсулирована в нем, называется...
28. Информация периода выполнения (RTTI) организована в виде нескольких уровней...

- 29. Средство проверки и приведения типов с использованием ключевых слов `is` и `as` – это...
- 30. Модель исключительных ситуаций в Object Pascal является...
- 31. Для обработки исключительных ситуаций в Object Pascal используется защищенный участок...
- 32. Для уверенности в том, что ресурсы, используемые вашим приложением, освободятся в любом случае, в Object Pascal используется защищенный участок...

Тема 2. Объектно-ориентированный подход

- 1. Методология программирования, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определенного класса, называется...
- 2. Методология проектирования, соединяющая в себе процесс объектной декомпозиции и приемы представления логической и физической, а также статической и динамической моделей проектируемой системы, называется...
- 3. Методология, при которой требования к системе воспринимаются с точки зрения классов и объектов, выявленных в предметной области, называется...
- 4. Абстрагирование относится к...
- 5. Инкапсуляция относится к...
- 6. Модульность относится к...
- 7. Иерархия относится к...
- 8. Типизация относится к...
- 9. Параллелизм относится к...
- 10. Сохраняемость относится к...
- 11. Отношение «`is a`» называется...
- 12. Отношение «`part of`» называется...
- 13. Операция, которая изменяет состояние объекта, называется...
- 14. Операция, которая считывает состояние объекта, но не меняет его, называется...
- 15. Операция, которая позволяет организовать доступ ко всем частям объекта в строго определенной последовательности, называется...
- 16. Между объектами могут быть такие отношения (указать все варианты)...
- 17. Между классами могут быть такие отношения (указать все варианты)...
- 18. Если отношение между классами характеризуется мощностью отношения, то оно называется...
- 19. Если отношение между классами изображается простым отрезком прямой линии, то оно называется...
- 20. Если отношение между классами изображается отрезком прямой линии с обычной стрелкой, то оно называется...

21. Если отношение между классами изображается отрезком прямой линии с закрашенным кружком, то оно называется...
22. Если отношение между классами изображается отрезком прямой линии с незакрашенным кружком, то оно называется...
23. Подходами к идентификации классов и объектов являются (указать все возможные варианты)...
24. Подход к анализу объектно-ориентированных систем, основанный на классической категоризации, называется...
25. Подход к анализу объектно-ориентированных систем, рассматривающий поведение как первоисточник классов и объектов, называется...
26. Подход к анализу объектно-ориентированных систем, основанный на мнении экспертов предметной области, называется...
27. Подход к анализу объектно-ориентированных систем, основанный на переборе сценариев классических подходов, поведения и предметной области, называется...
28. Подход к анализу объектно-ориентированных систем, основанный на «ручном» выстраивании иерархии классов в виде карточек специального вида, называется...
29. Подход к анализу объектно-ориентированных систем, основанный на описании задачи обычным языком с подчеркиванием существительных (будущих классов) и глаголов (будущих операций), называется...
30. Подход к анализу объектно-ориентированных систем, основанный на расширении имеющейся модели, описанной диаграммами потоков данных и некоторыми другими, называется...

Тема 3. Объектно-ориентированное программирование в среде C++

1. Задание данных вместе с функциями их обработки, что превращает их в новый тип данных «класс», называется...
2. Описание класса с дальнейшим его использованием для порождения иерархии классов с наследованием доступа каждого из них к коду и данным предка, называется...
3. Задание действию одного имени, которое передается вверх или вниз по иерархии классов, причем способ реализации действий для разных классов по иерархии может отличаться, называется...
4. Данные класса называются...
5. Функции класса называются...
6. Если класс объявляется вне любого блока, он называется...
7. Если класс объявляется внутри какого-либо блока, он называется...
8. Если тело метода определено внутри класса, то этот метод называется...
9. Метод, имя которого совпадает с именем класса, и который вызывается автоматически при создании объекта класса, называется...
10. Метод, имя которого содержит имя класса, и который вызывается автоматически при ликвидации объекта класса, называется...

11. Переменные типа «класс» называются (перечислить все возможные варианты)...
12. Для ссылок на элементы объекта внутри метода используется указатель...
13. Может ли конструктор возвращать значения?
14. Может ли класс иметь несколько конструкторов с разными параметрами для разных видов инициализации?
15. Может ли деструктор иметь аргументы и возвращать значения?
16. Элементы класса, доступные только в пределах этого класса, задаются с помощью спецификатора доступа...
17. Элементы класса, доступные в пределах этого класса и его потомков, задаются с помощью спецификатора доступа...
18. Общедоступные элементы класса задаются с помощью спецификатора доступа...
19. Наследование, при котором производный класс имеет одного родителя, называется...
20. Наследование, при котором производный класс имеет более одного родителя, называется...
21. Как описывается чистый виртуальный метод?
22. Класс, содержащий хотя бы один чисто виртуальный метод, называется...

СПИСОК РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ

1. **Мельников, А.Ю.** Программирование и алгоритмические языки : конспект лекций (для студентов направления «Системный анализ» всех форм обучения). Часть 2 / А.Ю. Мельников – Краматорск : ДГМА, 2012. – 33 с.
2. **Мельников, А.Ю.** Алгоритмизация и программирование : Учебное пособие для студентов специальности «Интеллектуальные системы принятия решений» / А.Ю. Мельников. – Издание 2-е, с изменениями. – Краматорск: ДГМА, 2010. – 96 с. – ISBN 978-966-379-437-2
3. **Мельников, А.Ю.** Работа в среде Borland-Delphi: Учебно-методическое пособие для студентов специальности «Экономическая кибернетика» / А.Ю. Мельников. – Краматорск : ДГМА, 2004. – 80 с. – ISBN 966-7851-63-X
4. **Мельников, А.Ю.** Работа в среде Lazarus : Учебное пособие / А.Ю. Мельников. – Краматорск : ДГМА, 2012. – 136 с. – ISBN 978-966-379-572-0.
5. **Мельников, А. Ю.** Объектно-ориентированный анализ и проектирование информационных систем : учебное пособие для сту-

дентов специальностей «Экономическая кибернетика» и «Интеллектуальные системы принятия решений» / А. Ю. Мельников. – Краматорск : ДГМА, 2006. – 184 с. – ISBN 966-379-103-9.

6. **Буч, Гради.** Объектно-ориентированный анализ и проектирование с примерами приложений на С++ / Гради Буч. – 2-е изд. / Пер. с англ. – М. : Изд-во «Бином»; СПб.: Невский диалект, 2001. – 560 с.

7. **Подбельский, В. В.** Язык С++ : учебное пособие / В. В. Подбельский. – 5-е изд. – М. : Финансы и статистика, 2001. – 560 с. – ISBN 5-279-02204-7.

8. **Страуструп, Б.** Язык программирования С++ / Б. Страуструп. – М. : Радио и связь, 1991. – 352 с.

9. **Павловская, Т. А.** С/С++. Программирование на языке высокого уровня / Т. А. Павловская. – СПб. : Питер, 2001. – 464 с. – ISBN 5-318-00001-0.

10. **Павловская, Т. А.** С/С++. Структурное программирование : практикум / Т. А. Павловская, Ю. А. Щупак. – СПб. : Питер, 2002. – 240 с. – ISBN 5-94723-447-5.

ПРИЛОЖЕНИЕ А

СПРАВОЧНЫЕ МАТЕРИАЛЫ

Палитра компонент среды визуального программирования,
основанной на языке Object Pascal



MainMenu – главное меню программы.



PopupMenu – всплывающее меню (появляется по щелчку правой кнопки мыши).



Label – отображение текста на экране. Можно изменять шрифт (свойство Font) и цвет (свойство Color) метки, а также надпись (свойство Caption); выравнивание строки производится свойством Alignment (при этом значение свойства AutoSize необходимо установить как False). Компонент *Метка* используется, как правило, в двух случаях: когда необходимо разместить на форме какую-либо поясняющую (или декоративную) надпись и для вывода результатов каких-либо расчетов. Следует заметить, что, в отличие от Турбо Паскаля, в Lazarus результаты расчетов нельзя просто так вывести на экран: все данные необходимо вводить и выводить с помощью компонент. Например, такой фрагмент программы:

```
x:=1.5; y:=Pi/2; z:=exp(x)+cos(y); Label2.Caption := FloatToStr(z);
```

позволит вывести в определенное место на экране рассчитанное значение переменной *z*



Edit – стандартный управляющий элемент для ввода и вывода строки текста. Он может быть использован для отображения короткого фрагмента текста и позволяет пользователю вводить текст во время выполнения программы. Вводимое значение имеет строковый тип (для присваивания числовым переменным необходимо соответствующее преобразование) и сохраняется в свойстве Text. Дополним немного предыдущий пример: представим, что значения *x* и *y* вводятся посредством двух полей ввода текста. Пример примет следующий вид:

```
x:=StrToFloat(Edit1.Text); y:=StrToFloat(Edit2.Text);  
z:=exp(x)+cos(y); Label2.Caption:=FloatToStr(z);
```

Таким образом, программа извлекает два числа, введенные пользователем в текстовые поля, производит некоторый расчет и заносит результат в текстовую



Memo – поле ввода многострочного текста. Вводимый текст сохраняется в свойстве `Lines`, которое, по сути, представляет массив строк (*нумерация в массиве начинается с нуля!*). Мемо-поле часто используется и для вывода данных: перед началом вывода поле можно очистить при помощи `Memo1.Lines.Clear`, а потом использовать `Memo1.Lines.Add (st)`, где `st` – подготовленная к выводу строка.



Button – компонента-действие, позволяющая выполнить что-либо при нажатии на ней во время выполнения программы.



ToggleBox – кнопка-флажок, позволяющая не только выполнить какое-либо действие, но и перевести эту компоненту в «нажатое» состояние.



CheckBox – включатель (флажок) – строка текста с маленьким окошком рядом, где можно поставить отметку, которая означает, что что-то выбрано. Если флажок «поднят», свойство `Checked` принимает значение `True`, «опущен» – `False`. Строка текста возле флажка определяется свойством `Caption`.



RadioButton – переключатель – позволяет выбрать только одну опцию из нескольких. Выбор одной компоненты автоматически снимает его с другой; имя свойства – такое же, как и у предыдущей компоненты – `Checked`.



ListBox – выбор из списка. Темы для выбора (строки) заносятся в массив `Items` (подобно `Lines` у `Memo`), индекс выбранной строки постоянно хранится в свойстве `ItemIndex` (отсчет начинается с нуля!). Если ничего не выбрано, `ItemIndex = -1`.



ComboBox – выбор из списка с возможностью ввода нового значения. Главное отличие от `ListBox` – результат выбора автоматически заносится в свойство `Text`. Если в поле ввести какой-нибудь текст, он будет присвоен свойству `Text`, однако индекс выбора останется равным `-1`.



ScrollBar – полоса прокрутки (появляется автоматически в объектах редактирования при необходимости прокрутки текста для просмотра). Минимальное значение «бегунка» задается свойством `Min`, максимальное – `Max`, текущее – `Position` (всегда $Min \leq Position \leq Max$). Изменение положения «бегунка» вызывает событие `Change`.



RadioGroup – группа переключателей. Компонента аналогична размещению нескольких компонент *RadioButton*, но обладает рядом преимуществ. Так, количество строк для выбора и их содержимое можно изменять динамически, в ходе работы программы. Свойство *Items* содержит список строк с заголовками элементов, номер выбранного элемента заносится в *ItemIndex*; при отсутствии выбора *ItemIndex* = -1. Также при помощи изменения свойства *Columns* можно размещать кнопки в несколько столбцов.



CheckGroup – группа флажков. Компонента аналогична размещению нескольких компонент *CheckBox*, но количество строк для выбора и их содержимое можно изменять динамически, в ходе работы программы. Свойство *Items* содержит список строк с заголовками элементов, сами флажки (выбор) сохраняются в массиве *checked*.



Panel – контейнер общего назначения, похожий на *GroupBox*, используется в декоративных целях.



BitBtn – кнопка с изображением. Картинку можно загрузить самому (свойство *Glyph*) или использовать predefined типы (свойство *Kind*), при выборе которых кнопка принимает соответствующий вид (*bkOk*, *bkClose*, *bkAbort* и т. д.)



SpeedButton – кнопка панели инструментов. Отличается от предыдущей невозможностью сохранения фокуса. То есть, если, находясь в поле ввода текста, нажать такую кнопку, указатель останется в текстовой строке, хотя действие, связанное с кнопкой, начнет выполняться.



StaticText – текстовая метка. Аналог *Label* с некоторыми добавлениями: например, рамкой, рельефностью и т. д.



Image – отображение графического изображения на форме (за само изображение отвечает свойство *Picture*). Воспринимает форматы JPG, BMP, PNG, ICO и еще несколько (GIF и TIFF при этом не поддерживаются). Если картинку подключить во время дизайна программы, то она добавится к EXE файлу. В случае большого размера рисунка его целесообразно загружать во время выполнения приложения, например: *Image1.Picture.LoadFromFile ('mypict.bmp')*. Свойство *Center* размещает рисунок в центре области *Image*. Свойство *Stretch* позволяет автоматически масштабировать рисунок так, чтобы он занял всю рабочую область *Image*



Shape – простейшая геометрическая фигура: прямоугольник (`stRectangle`), квадрат (`stSquare`), скругленный прямоугольник (`stRoundRect`), скругленный квадрат (`stRoundSquare`), эллипс (`stEllipse`), окружность (`stCircle`). Вид фигуры задается свойством `Shape`; изменение этого свойства приводит к немедленной перерисовке изображения.



PaintBox – место для рисования (невизуальная компонента). Рисовать можно при помощи специального свойства `Canvas` и свойств `Font`, `Pen` и `Brush`.



NoteBook – компонента для создания многостраничного интерфейса. В начале работы с этой компонентой необходимо создать нужное число страниц («Добавить страницу» во всплывающем меню) и ввести заголовки каждой страницы (свойства `Caption`). Каждая страница будет содержать свой список компонент – таким образом, использование *NoteBook* является альтернативой многооконному интерфейсу.



LabeledEdit – поле ввода текста с дополнительной строкой-заголовком. Место этой строки определяется свойством `LabelPosition` (сверху, снизу, слева, справа), все ее свойства – объектом `EditLabel`.



CheckListBox – группа флажков. Представляет комбинацию списка `Listbox` и массива флажков `CheckBox`. Соответственно, имеет свойства `Items` (строки для выбора) и `ItemIndex` (номер выбранной строки), однако также содержит массив `Checked`.



ScrollBox – панель с полосами прокрутки в вертикальном и горизонтальном направлениях. Позволяет создать на форме прокручиваемую область с размерами большими, нежели сама форма, где можно разместить свои объекты.



StringGrid – таблица, компонента для ввода или вывода табличной информации. Число строк и столбцов задается свойствами `RowCount` и `ColCount` соответственно; можно определить «фиксированные» строки и столбцы (свойства `FixedRows` и `FixedCols`): данные этих строк выделяются серым цветом и не подлежат изменению (фактически это заголовки). Чтобы пользователь имел возможность изменять данные в ходе работы программы, надо установить свойство `goEditing` как `True` (по умолчанию установлено `False`). Доступ к содержимому ячеек осуществляется свой-

ством `Cells[Acol,Arow]`, где `Acol` и `Arow` – это индексы столбца и строки соответственно (обратите внимание: именно в таком порядке!). Ячейка рассматривается как текстовая строка, так что для занесения в нее числа его необходимо предварительно преобразовать. Ширина столбцов задается через `DefaultColWidth` (по умолчанию для всех) или массив `ColWidths[Acol]`. Высота строк задается через `DefaultRowHeight` (по умолчанию для всех) или массив `RowHeights[Arow]`. При «движении» по таблице в ходе работы программы текущие координаты заносятся в свойства `Row` (номер строки) и `Col` (номер столбца). Во всех случаях нумерация начинается с 0.



ColorBox – выбор цвета из выпадающего списка. По сути, представляет собой `ComboBox` с заранее введенными значениями массива `Items`.



TrackBar – ползунок. Подобно компоненте `ScrollBar` позволяет устанавливать минимальное (`min`), максимальное (`max`) и текущее (`position`) значения, изменять расположение (горизонтальное или вертикальное).



ProgressBar – индикатор процесса. Отображает ход выполнения длительного во времени процесса. Во многом похож на `ScrollBar` и `TrackBar`, однако с его помощью можно только отображать величину, а не изменять.



TreeView – иерархическое дерево. Служит для показа ветвящихся структур. Содержит связанные узлы (свойство `Items`), каждый из которых может содержать значок, текст и произвольный объект; любой узел может иметь собственный список дочерних узлов.



ListView – иерархический список. Имеет два существенных отличия от `TreeView`: с одной стороны, допускается не более одного уровня вложенности элементов, однако сами элементы могут показываться в одной или нескольких колонках, с крупными или мелкими значками.



StatusBar – строка состояния. Обычно располагается в нижней части формы; может иметь несколько панелей.



Timer – таймер. Событие `OnTimer` периодически вызывается через промежуток времени (в миллисекундах), указанный в свойстве `Interval`. Реально период времени не может быть меньше 10 мс (длительность так называемого «системного тика» в ОС Windows), но ошибки не возникнет в любом случае.