МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение высшего образования «Казанский национальный исследовательский технический университет им. А.Н. Туполева-КАИ»

Альметьевский филиал

«ПРОГРАММИРОВАНИЕ»

Методические указания по написанию курсовых работ

для направления подготовки

09.04.01 «Информатика и вычислительная техника»

Альметьевск 2022

СОДЕРЖАНИЕ

Цель курсовой работы: получить практические навыки в создании приложений с использованием высокоуровневых методов программирования в среде Microsoft Visual Studio, язык программирования C#.

Порядок выполнения работы

1. Взять у преподавателя вариант задания.
2. Изучить методическое указание.
3. Познакомиться со страницей, содержащей предполагаемые структуру и содержание курсовой работы.
4. Познакомиться с графиком выполнения курсовой работы.
5. В папке для курсовой работы создать отдельный файл для хранения пояснительной записки к курсовой работе. В нем должны помещаться фрагменты, из которых в дальнейшем будет образован полный текст пояснительной записки. Первыми страницами в нем должны быть: титульный лист, лист задания, список литературы. Рекомендуется сразу выставить по тексту названия глав и параграфов в нужном формате. В конце периода из совокупности созданных фрагментов должен сложиться весь отчет – пояснительная записка.
6. Предусмотреть ***постоянное копирование документа на другие носители*** для уменьшения потерь в случае повреждения и удаления папок.
7. Распечатать пояснительную записку и сдать ее для проверки.
8. Защитить работу перед комиссией.

Требования

1. Исходные данные изначально следует разместить в файле данных.
2. Разрабатываемое приложение должно содержать не менее трех форм: главная форма; форма, на которой отражаются результаты решения задачи; форма, содержащая информацию об авторе, включая по возможности его фотоизображение.
3. В проекте должен быть предусмотрен отдельный модуль, в котором должен быть размещен созданный класс (АТД).
4. Изготавливаемый программный продукт должен быть откомментирован, написан в соответствии с требованиями структурного программирования.
5. Для защиты курсовой работы необходимо подготовить презентацию, в которую следует включить следующие слайды: постановка задачи, интерфейс класса, интерфейсы с результатами, блок-схемы наиболее сложных алгоритмов, выводы.

Рекомендации

В методическом указании имеется большая глава: «Содержательное наполнение разделов пояснительной записки». Эта глава является основной, и она состоит из разделов, наименования которых совпадают с наименованиями параграфов, отраженных в “Структуре и содержании курсовой работы”.

Содержательная часть этих разделов предназначена для того, чтобы помочь в написании текста пояснительной записки. Каждый раздел в общем случае представлен тремя компонентами:

а) описанием того, какую информацию и каким образом следует отразить в соответствующем параграфе пояснительной записки;

б) пример описания;

в) указание литературного источника, где можно получить дополнительную помощь.

Методические указания

Как это следует из цели, поставленной в курсовой работе, основная задача, стоящая перед студентом курса: научиться составлять программы высокого качества. Такие программы должны быть легко модифицируемыми, простыми в обращении. Они должны быть написаны с использованием современных методов программирования, таких как *ООП* (объектно-ориентированные программирование), модульное программирование, процедурное программирование, визуальное программирование, событийное программирование.

Как известно, класс – это определяемый пользователем тип, объединяющий в себе группу данных и функций для работы с этими данными. В определении нового типа всегда лежит идея – отделить (абстрагироваться) несущественные подробности реализации от тех качеств, которые существенны для его правильного использования.

Одно из мощных преимуществ классов, как типов данных, заключается в том, что классам присуща структура, позволяющая моделировать реальные объекты. Любой предмет может быть описан набором своих характеристик, т.е. данных. Работать с моделью реального мира тем проще, чем больше отношения между данными в модели объекта напоминают отношения между характеристиками этого объекта.

Моделирование объектов в программе также называется *абстракцией*. Речь идет об имитации реально существующих объектов, отражающей особенности их взаимодействия в окружающем мире. А концепции виртуальной реальности выводят принцип абстракции на совершенно новый уровень, не связанный с физическими объектами. Абстракция необходима, потому что успешное использование ООП возможно лишь в том случае, если вы сможете выделить содержательные аспекты своей проблемы.

Поэтому основу решения задачи должно составлять разработка класса. Именно вопросам класса или вопросам по созданию АТД, подготовки методов класса отведено максимальное время курсовой работы. Приступая к построению объектной модели, всегда задавайте себе вопрос: какие свойства и методы должны входить в объект, чтобы он адекватно моделировал ситуацию для решения поставленной задачи?

Поскольку по условиям проекта приложение должно быть подготовлено средствами C++ Builder, студент должен овладеть методами *модульного программирования*, визуального программирования, событийного программирования.

Структура и содержание курсовой работы

Титульный лист.

Аннотация.

Задание.

Оглавление.

Введение.

1. Основная часть.

1.1. Постановка задачи

1.2. Анализ и исследование задачи, построение модели

1.3. Разработка классов.

1.4. Формализация расчетов.

1.4.1. Алгоритмы методов класса.

1.4.2. Алгоритмы, обеспечивающие функциональность приложения.

2. Разработка структурной схемы интерфейса

3. Разработка схемы связности модулей

4. Схема движения информационных потоков

5. Тестирование программы

5.1. Разработка плана тестирования.

5.2. Оценка результатов проведения тестирования

6. Заключение

7. Список литературы

8. Приложения

Содержательное наполнение разделов пояснительной записки

Аннотация

Содержит перечень используемых ключевых слов, очень краткое[1] содержание работы, число страниц пояснительной записки, число рисунков, таблиц, приложений.

[1] 2-3 предложения

Введение

Введение должно содержать общие сведения по теме курсовой работы. Так, если в основе работы лежат списки, то требуется дать информацию о списках (что это, зачем, особенности и т.д.). Если речь идет о множества, то сведения о множествах и т.д.

Во введении также необходимо отразить:

- актуальность выбранной темы (например, сказать, что в современном обществе или в деятельности любого современного предприятия информация является одним из важнейших ресурсов, выделяясь в самостоятельный фактор для принятия решений, и от средств ее обработки во многом зависит эффективность принятия решений);
- цель (например, приобрести навыки в создании АТД и разработке приложений в среде C#с применением современных технологий программирования);
- задачи[2], решаемые в проекте (это может быть построение математической
- используемые модели программирования (например, императивное программирование, модульное программирование, структурное программирование, объектно-ориентированное программирование, основанное на классах…)
- модели[3], создание класса, использование экземпляров класса для принятия решений, …);
- практическую значимость полученных результатов (где можно использовать);
- какого рода ресурсы необходимы для реализации (ПК, программное обеспечение…уточнить какое);
- перспективы совершенствования изготавливаемого программного продукта.

Постановка задачи

Составляющие элементы этого раздела м.б. следующие:

- формулировка условия задачи;
- сбор информации о задаче и выделение физического объекта;
- определение конечных целей решения задачи;
- определение формы выдачи результатов;
- описание данных (их типов, диапазонов величин, структуры и т.п.).

[2,3] Уточнить, каких в соответствии с темой

Формулировка условия задачи ***выдается преподавателем***. В качестве примера для объяснения хода и логической основы некоторых элементов выполнения работы предлагается к рассмотрению следующая постановка:

О товаре, размещенном на складе, имеется информация вида: номер артикула, наименование товара, общее кол-во, выделенное кол-во, не поставленное количество, цена единицы. Разработать приложение (программу), которое выдает информацию о товаре по требованию пользователя и подводит итоги по не поставке товаров.

Сбор информации о задаче и выделение физического объекта. Следует определиться, с какими данными разработчик проекта имеет дело. Так как в задаче речь идет о товаре, размещенном на складе, нужно указать, какой товар (перечислить его: стол, стул, …), что связано с поставкой/непоставкой (М.б. документы на заказ и на доставку). Выделить физический объект (множество товаров) и определить его свойства. В том числе, количество элементов – единиц товара. А т.к. в данном случае физический объект определен как множество, то следует выделить свойства отдельного представителя из этого множества. При этом необходимо выявить самые существенные свойства, необходимые для решения задачи. Выделив наиболее важные факторы, можно пренебречь менее существенными. Параметрами отдельного представителя (единицы товара), например, могут быть: наименование, цена, фирма-изготовитель, поставщик,количество).

Определение конечных целей решения задачи. В соответствии с условием данной задачи разрабатываемая программа должна предоставлять пользователю хранить информацию о товарах и формировать из нее нужную информацию по его требованию. Такими требованиями могут быть: получить список товаров заданного наименования с его характеристиками (какими?), список поставщиков, …

Определение формы выдачи результатов. Например, представить список поставщиков в виде таблицы, в которой можно было бы увидеть наименование поставляемого им товара …

Описание данных (их типов, диапазонов величин, структуры и т.п.). Например, наименование товара представляется строкой символов, количество символов не превышает

20. Цена указывается в рублях, может быть представлена вещественным числом, диапазон ценовых колебаний в промежутке от 10000 до 500...

Здесь целесообразно подготовить для дальнейшей работы текстовый файл с реальными данными.

Анализ и исследование задачи, построение модели

Составляющими данной главы м.б. следующие элементы:

- выделение математического объекта;

- анализ существующих аналогов;
- анализ технических и программных средств;
- разработка математической модели;
- разработка требований к приложению.

Выделение математического объекта. Сказать, например, что для того, чтобы иметь возможность хранить и обрабатывать данные о физическом объекте, эти данные нужно представить в виде, приспособленном для обработки математическими методами. Для этого нужно перейти от физического объекта к объекту математическому. В нашем случае нужно перейти от физического объекта — множество товаров, где каждый товар имеет свои физические характеристики, к математическому объекту, описывающему это множество. Причем, каждый элемент математического множества должен быть представлен эквивалентными свойствами элемента физического множества (имеет структуру из свойств).

Далее можно сказать о том, что для описания математического множества можно воспользоваться таким математическим объектом, как «массив». В данном случае это будет массив структур.

Анализ существующих аналогов. Сказать, что хранить и обрабатывать массив структур, описанных данных можно разными способами, например, средствами базы данных. Привести примеры известных баз, например, Access (особенности…), в Excel (особенности…).

Анализ технических и программных средств. ПК – техническое средство. Объяснить, почему выбрана среда C#, а не Excel или др. (возможности ООП и визуального…).

Разработка математической модели.Под математической моделью понимают систему математических соотношений – формул, уравнений, неравенств и т.д., отражающих существенные свойства объекта. Метод математического моделирования сводит исследование поведения объекта или его свойств к математическим задачам. Следует выписать формулы, например, с использованием знаков суммирования …

При этом можно пользоваться и такой схемой действий:

1. выделить предположения, на которых будет основываться математическая модель. Например, предположить, что информация о товаре будет сосредоточена в структурах данных (или содержатся в файле, или в таблице, или…);
2. определить, что считать исходными данными и результатами;
3. записать математические соотношения, связывающие результаты с исходными данными. Какие методы обработки (с описанием подхода к решению, например,

«... решение сводится к задаче накопления суммы элементов… »)

Выведенные зависимости и формулы в общем случае могут быть представлены уравнениями, системами уравнений – линейных, интегральных, матричных, дифференциальных и

др.. На этом этапе для нахождения решения также строится неформальный алгоритм (производная от длины и решение уравнения: производная равна 0, …).

При построении математических моделей далеко не всегда удается найти формулы, явно выражающие искомые величины через данные. В таких случаях используются математические методы, позволяющие дать ответы той или иной степени точности.

Разработка требований к приложению. Например, сказать о том, что пользователь должен иметь возможность для задания исходных данных вручную, возможность для сохранения данных в файле, возможность формировать файл данных с целью повторного использования; извлекать данные из файла данных; выбирать вид отображаемой информации (например, общее количество товара, суммарная стоимость всего товара, номенклатура товара и т.д. в соответствии с требованиями); отображать нужную информацию в соответствии с выбранным видом …, а также обеспечение соединений этих данных в соответствии с пользовательской логикой (все исходя из поставленной задачи и личных мотивов).

Разработка классов

Следует сказать о том, что для решения задачи на машине следует от математического объекта перейти к программному объекту. Таким программным объектом будет являться класс, определенный пользователем. Класс по своей сути представляет запись математической модели на языке программирования.

Разработку классов рекомендуется проводить в следующем порядке:

- разработка структуры данных, помещаемых в поля класса;
- разработка модели поведения математического объекта;
- разработка схемы иерархии классов;
- разработка интерфейсов классов.

Разработка структуры данных, помещаемых в поля класса. Здесь следует отметить, что математический объект «массив структур» определяется двумя характеристиками: именем этого массива и его размерностью — *количеством элементов*. Все остальные свойства следуют однозначно из этих двух его характеристик.

А т.к. структура элемента/представителя массива имеет набор характеристик, представленных разными типами, имеет смысл объединить характеристики, описывающие отдельный товар, в единое целое.

Для этого в C++ предусмотрен тип struct — структурный тип.

Далее следует расположить интерфейс структуры, у которой должны быть осмысленные имена, записанные на английском языке. Поля структуры следует снабдить комментариями. Как

это показано на листинге 1.

Листинге 1. Интерфейс структуры Ware – «товар»

```
struct Ware
{
char Mark [10]; //Марка товара
char NameWare [22];//Наименование товараint
Quantity; //Количество единиц товара
……
float price;//Цена товара
};
```

Следует отметить, что элементы массива хранятся в массиве, память для которого должна будет выделяться автоматически в момент создания объекта/экземпляра типа класса. Именно благодаря такому решению можно будет создавать объекты – массивы с любым количеством элементов. Ключевой момент в том, что массив структур в *C++* в таком случае должен быть описан как *указатель*, например, типа Ware *. Эти две основные характеристики массива и будут определять поля класса.

Разработка модели поведения математического объекта. Под моделью поведения, а эта терминология принята в технологии ООП, принято понимать совокупность методов, необходимых для решения, определенного в классе, набора задач. Класс должен предоставлять пользователям следующие возможности: решать задачи, заявленные в условии: (перечислить); задавать исходные данные (что и как); генерировать значения по какому-либо правилу (как); формировать файл с целью повторного использования (…); извлекать данные (из файла, …) данных и т.д. Сформулированные выше возможности будут определять поведение объектов класса и должны быть заложены в методы класса. Естественно, что кроме заявленных выше методов класс-вектор, как и любой другой класс, должен иметь несколько конструкторов и деструктор.

Разработка схемы иерархии классов. При проектировании классов, как и при создании любой программы, разработчик должен изначально исходить из предположения, что предложенная задача со временем может претерпевать изменения, связанные с возрастанием и изменением запросов пользователя. Поэтому необходимо предусмотреть механизм, который бы позволял мобильно изменять интерфейсы классов, не затрагивая их реализацию. С этой целью имеет смысл сразу проектировать семейства классов.

Проектируя семейство классов, необходимо распределить, какие методы будут реализованы в базовом классе и какие из них будут абстрактными; сколько необходимо потомков и каким набором методов и свойств они должны обладать.

В рамках поставленной задачи целесообразно выделить семейство из трех классов. В базовый класс поместить методы, общие для перечисленных задач, в класс – наследник первой очереди поместить методы решения этих перечисленных задач. А в третий класс – наследник от наследника поместить методы, связанные с вводом и отображением данных. В таком представлении первые два класса будут мобильными, то есть они практически без исправлений могут компилироваться под любой системой программирования для C++ и для различных операционных систем. Третий класс будет содержать методы с учетом работы в конкретной системе программирования. То есть он будет дополнять предыдущие классы методами, которые привязаны к типовым элементам, таким как классы и компоненты, предопределенные в нашем случае в среде C++ Builder.

Разработка интерфейсов классов. В этом разделе следует на листингах представить интерфейсы классов, сопроводив их пояснениями, можно в следующей последовательности:

характеристика класса. (Например, «…среди методов класса два конструктора и один деструктор и методы, предназначенные… Поля класса … перечислены в секции protected. Эта метка открывает эти поля для доступа к ним со стороны будущих наследников. Такое решение позволяет выявить ошибки типа нарушения прав доступа к данным еще на этапе компиляции программы. Методы класса перечислены в секции public, т.к. они должны быть доступны из основной программы. Один конструктор будет использоваться для работы с объектами классов в случае ручного ввода данных, а второй – в случае ввода данныхиз файла…»);

- листинг с интерфейсом;
- спецификации методов класса.

На данном этапе разработки проектировать поведение методов класса следует по принципу «черного ящика»: объект реагирует на входные параметры, результаты являются выходными параметрами, а фактическая реализация остается неизвестной. Причем, входные и выходные параметры являются либо полями класса, либо параметрами метода.

Следует иметь в виду, что все имена: класса, полей, методов должны быть осмысленными, начинаться с заглавной буквы, записаны на английском языке (не следует использовать русские слова с английскими буквами).

Формализация расчетов

В данном разделе следует поместить описания методов классов по схеме:

1. Заголовок, в котором слова о назначении метода.
2. Прототип.
3. Словесное описание алгоритма в общем виде (суть действий).

4. Детальное описание алгоритма.

Ниже приведены примеры описаний.

Пример 1.

Конструктор с одним параметром – «имя файла»

```
TArray(char* NameFile);
```

Конструктор формирует экземпляр класса на основании той информации, которая содержится внутри файла, имя которого передается через параметр конструктора NameFile. Следовательно, прежде чем воспользоваться услугами такого конструктора, следует создать файл и подготовить его соответствующим образом. В данном случае – это текстовый файл, в котором содержатся числа, представляющие собой значения координат вектора. Причем, в самом начале этого файла записано целое число, предопределяющее количество следующих значений, т.е. размерность вектора. Обычно это количество располагают в отдельной строчке.

Алгоритм конструктора с параметром «имя файла» построен по следующей схеме: а) обнуляются поля класса на случай, если неудача с файлом;

б) открывается файл для чтения;

в) проверка, если файл не существует, то – выход;

г) проверка, если файл пуст, то – выход;

д) читается размер и инициализируется поле FCount;

е) отводится память под массив и инициализируется указатель FData;ж) вцикле по числу элементов читаются из файла данные в массив;

з) закрывается файл.

Описание конструктора представлено в листинге 2.

Листинг 2. Описание конструктора, формирующего объект на основе информации, содержащейся в файле данных

```
TArray::TArray(char* NameFile)
// Конструктор массива с параметром «имя файла»
{
// на случай, если ошибка чтения (нет файла)
FCount = 0; FData = NULL;
// открыть файл NameFile для чтения
ifstream iFin(NameFile, ios::in);


if(!iFin)return; // если нет файла, то – выход if(iFin.eof())return; // если файл пуст, то - выходiFin>>FCount;
        // прочитать число элементов
FData = new int[FCount];  // отвести память под массив
```

```
// прочитать данные из файла
for (int i = 0; i < FCount; i++) iFin>>FData[i];


iFin.close();  // закрыть файл
}
```

Пример 2.

Переопределение длины вектора

```
void SetCount (int aCount);
```

Метод используется в том случае, когда в процессе преобразований или обработки данных изменяются размеры массива: либо происходит увеличение числа элементов, либо уменьшение. Тогда для корректной работы с экземпляром класса нужно переопределить его параметры. Метод имеет один параметр, который передает новый размер массива. Схема действий алгоритма по переопределению полей класса можно описать следующими шагами:

а) проверяется размеры нового и старого массивов на совпадение;

б) если размеры совпадают, то ничего менять не нужно;

в) если размеры не совпадают, то:

- создается вспомогательный массив;
- копируются данные из старого массива во вспомогательный массив с целью их сохранения;
- освобождается память занятая старым массивом;
- указатель на массив, переназначается на вспомогательный;
- количество элементов устанавливается равным новому значению,переданному через входной параметр.

Описание метода SetCount, предназначенного для переопределения параметров класса, представлено в листинге 3.

Листинг 3. Описание метода SetCount, предназначенного для переопределения длины вектора

```
void TVector::SetCount (int aCount)
// Ввести-изменить размер вектора на aCount
// Если размеры новый и старый не совпадают, то
{
if (aCount <= 0)
{FCount = 0;
```

```
delete []FData; FData = NULL;
}
if (FCount != aCount)
{
int *aux = new int [aCount];
for (int i = 0; i < min (FCount, aCount); i++)FData [i] = aux [i];
delete []FData;
FData = aux;  // адрес нового массива в FCount
FCount = aCount;
}
return;
}
```

Замечание: к 2-3 методам обязательно добавить блок-схему!

Разработка структурной схемы интерфейса

Разработчик должен предоставить ответы на следующие вопросы:

- Какие требования выставляются к интерфейсу?
- Почему именно такой интерфейс?
- Есть ли другие варианты?

На этом этапе разрабатывается структурная схема интерфейса программы и детали управления – логика (текстовое пояснение) решения задачи в программе. Разработка структурной схемы позволяет выверить все детали проекта, определить взаимоотношения (текстовое пояснение) между отдельными частями программы. Разработка структурной схемы определяет (текстовое пояснение) содержание программных сообщений. На основании этой схемы в дальнейшем строится *схема движения информационных* потоков ~~и~~ т.д. Можно начать фразой: «Существует четыре основных (все остальные – производные) критерия оценки качества любого интерфейса, а именно:

- скорость работы пользователей;
- количество человеческих ошибок;
- скорость обучения;
- субъективное удовлетворение.

Скорость выполнения работы является важным критерием эффективности интерфейса. Длительность выполнения работы пользователем состоит из длительности восприятия исходной информации, длительности интеллектуальной работы (пользователь думает, что он должен сделать), длительности физических действий пользователя и длительности реакции системы. Как правило, длительность реакции системы является наименее значимым фактором.

Согласно Дональду Норману, взаимодействие пользователя с системой (не только компьютерной) состоит из семи шагов:

- формирование цели действий;
- определение общей направленности действий;
- определение конкретных действий;
- выполнение действий;
- восприятие нового состояния системы;
- интерпретация состояния системы;
- оценка результата.

Из этого списка становится видно, что процесс размышления занимает почти все время, в течение которого пользователь работает с компьютером, во всяком случае, шесть из семи этапов полностью заняты умственной деятельностью. Соответственно, повышение скорости этих

размышлений приводит к существенному улучшению скорости работы.

К сожалению, существенно повысить скорость собственно мышления пользователей невозможно. Тем не менее, уменьшить влияние факторов, усложняющих процесс мышления, вполне возможно.

Пользователь должен знать:

- что он хочет получить на выходе (решение задачи);
- как минимум одну последовательность действий, приводящую к успешному результату;
- где ему найти все объекты, участвующие в процедуре решения;
- как определять пригодность объектов для их использования;
- как управляться с объектами.

Список, как видим, довольно внушительный. И если с первым пунктом проблем обычно не возникает, то остальные требуют определенных усилий. А помочь разобраться в том должен интерфейс, с встроенной системой подсказок действий. Следовательно, должен быть продуман механизм *управления программой через элементы интерфейса*.

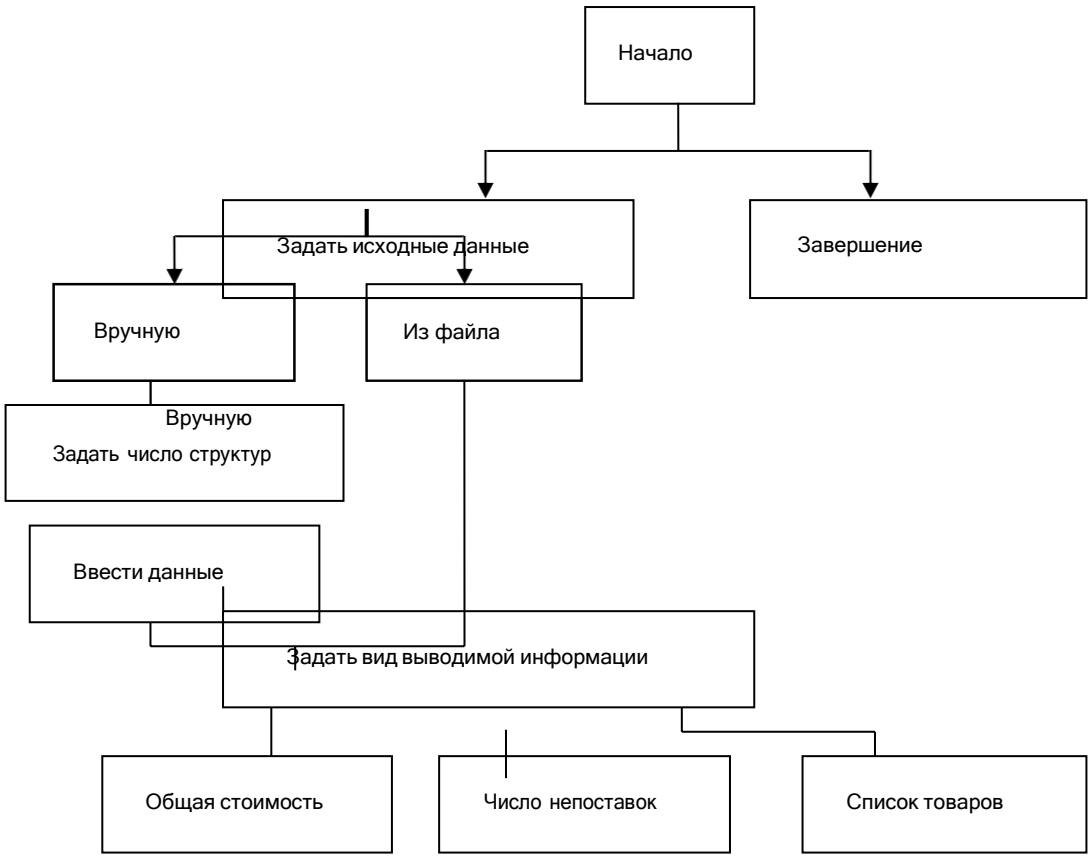

Рис.3. Структурная схема интерфейса

Структурную схему обязательно согласовать с преподавателем!

В описательной части следует отметить, что в соответствии с проведенным выше анализом

задачи разрабатываемая программа должна содержать: последовательные задания исходных данных (каких) для эксперимента, определяющих условия решения; выбор вида решения (расписать) и вида результатов (охарактеризовать в соответствии с ранее определенными функциями). В результате может быть сформирована структурная схема функционирования интерфейса подобная тому, как это представлено на рис.3.

Пояснить, что структурная схема интерфейса представляет собой графическую интерпретацию конструкции диалога, задающей требуемую последовательность обменов данными между пользователем и системой. Пояснить, что в каждом состоянии (каком) диалога разрабатываемая система ожидает ввода сообщения – реакции (уточнить) от пользователя и в зависимости от введенной информации переходит в другое состояние (какое). Пояснить, что при завершении диалога осуществляется соответствующая обработка данных (какая) и выдается определенная информация (какая) на экран.

Описание интерфейса приложения

Этот раздел может начинаться словами: “Для решения поставленной задачи в соответствии с разработанной схемой было создано приложение, интерфейс которого включает в себя N[4] форм. Далее следует представить рисунки с изображением форм + описание деталей соответствующей формы (со ссылками на структурную схему) + комментарии к деталямформы”.

Описание структуры приложения и схема связности модулей

Можно начать словами: “Функционирование программ приложения формируется на базе следующихмодулей:

Project.dpr– главный модуль, процедурами которого являются следующие модули: Unit1.pas,MyUnit.pas, …

Unit1.dfm, Unit1cpp – модули, соответствующий главной форме Form1, с помощью которой осуществляется презентация данного программного продукта, а также вызов модулей: MyUnit.cpp, …

MyUnit.cpp – модуль, содержащий классы для решения задачи (реализующий класс (имя класса) и егоподкласс (наследник) (имя)).

Между модульной структурой и структурами данных существует связь, которая может бытьпредставлена в виде следующей схемы (рис.5).

[4] N – равно числу созданных форм, обеспечивающих функционирование приложения

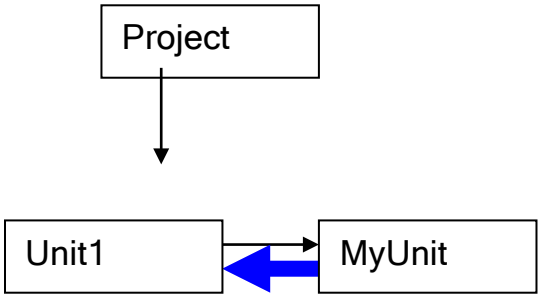

Рис.5. Схема связности модулей

Схема движения информационных потоков (связать с экземпляром класса)

Функционирование любой приложения (программы) можно рассматривать, как обработку некоторого входного потока данных. Данные поступают от входа, преобразуются по правилам решения задачи, и в преобразованном виде передаются к выходу внешнимпользователям.

Задача анализа решения состоит в установлении правильности обработки данных. Отсюда, задача определения способа хранения, порядка и правил организации данных,

является очень важным элементом при разработке приложения. Четкое структурирование данных, выполняемое в процессе разработки, способствует уменьшению сложности системы и снижает вероятность ошибок из-за их неправильного использования.

Существуют различные способы организации и хранения информационных объектов: файлы, списки, строки, массивы, таблицы, структуры данных, классы, представляющие собой организованные наборы записей данных с методами их обработки и т.д.

Ряд данных, используемых многими модулями и группами системы, определяются как глобальные (файлы, структуры, классы, …). Это переменные, характеризуемые наиболее широким использованием и соответствующие высшему иерархическому уровню среди данных. Все эти данные в интерфейсе приложения отображаются с помощью тех или других визуальных компонент в определенной последовательности и в соответствии с выполненным преобразованием.

Пример описания: “*В интерфейсе приложения были определены следующие информационные объекты:*

1. Файл, содержащий исходные данные, который имеет следующую структуру: …(описание структуры файла).

2. Компонент Edit1, позволяющий создать однострочное текстовое поле, предназначенное длязадания числа строк в матрице.

3. Компонент Edit2, позволяющий создать однострочное текстовое поле, предназначенное длязадания числа столбцов в матрице.

4. Компонент StringGrid1, позволяющий задать элементы матрицы.

5. Компонент Edit3, позволяющий создать однострочное текстовое поле, предназначенное дляотображения на форме результата

6. …

Привести схему движения информационных потоков

Тестирование программы

Тестирование созданного программного продукта следует начинать с разработки плана тестирования. При этом следует помнить, что весь процесс тестирования можно разделить на три этапа:

- Проверка в нормальных условиях. Предполагает тестирование на основе данных, которые характерны для реальных условий функционирования программы.
- Проверка в экстремальных условиях. Тестовые данные включают граничные значения области изменения входных переменных, которые должны восприниматься программой как правильные данные. Типичными примерами таких значений являются очень маленькие или очень большие числа и отсутствие данных. Еще один тип экстремальных условий – это граничные объемы данных, когда массивы состоят из слишком малого или слишком большого числа элементов.
- Проверка в исключительных ситуациях. Проводится с использованием данных, значения которых лежат за пределами допустимой области изменений.

Известно, что все программы разрабатываются в расчете на обработку какого-то ограниченного набора данных. Поэтому важно получить ответ на следующие вопросы:

- Что произойдет, если в программе, не рассчитанной на обработку отрицательных и нулевых значений переменных, в результате какой-либо ошибки придется иметь дело как раз с такими данными?
- Как будет вести себя программа, работающая с массивами, если количество их элементов превысит величину, указанную в объявлении массива?
- Что произойдет, если числа будут слишком малыми или слишком большими?

Наихудшая ситуация складывается тогда, когда *программа воспринимает неверные данные как правильные и выдает неверный, но правдоподобный результат*. Программа должна сама отвергать любые данные, которые она не в состоянии обрабатывать правильно. Следует

попытаться представить возможные типы ошибок, которые пользователь способен допустить при работе с Вашей программой и которые могут иметь неприятные для нее последствия. Нужно не забывать, что способ мышления пользователя отличается от способа мышления программиста. Если помнить об этом, то нужно предусмотреть обработку ошибок типа: отсутствие нужного файла, неправильные форматы данных и т.д. Список действий, которые могут привести к неправильному функционированию программы, довольно длинный и зависит от того, что делает приложение в данный момент времени.

Основной смысл этого этапа состоит в проверке того, насколько программный продукт в том виде, в котором он получился, соответствует требованиям, установленным в процессе согласования спецификации. Каждая функция или метод класса должны соответствовать требованиям, определенным для них на этапе спецификации.

Разработка плана тестирования состоит из следующих шагов:

1) определение последовательности действий, которые позволяют проверить как работу отдельных методов, так и их совокупности;

2) подготовка входных данных, для которых известны результаты тестирования;

3) определение места расположения тестовых данных.

Разработка алгоритма процедуры тестирования состоит в разработке процедуры в соответствии с составленным выше планом тестирования. Эту процедуру можно представить блок-схемой с соответствующими комментариями.

Оценка результатов тестирования – содержит сравнение выходных данных работы отдельных процедур с контрольными значениями, которые получены в результате выполнения процедуры тестирования.

Заключение

Подводятся общие итоги проделанной работы, дается их оценка, делаются общие выводы.

Например, начало может быть следующим:

“В процессе разработки программного приложения (программной системы) для решения конкретной задачи было проделано следующее:

1. изучена литература по теме проекта;
2. разработан алгоритм решения задачи по обработке …;
3. разработан интерфейс приложения;
4. разработана схема движения информационных потоков;
5. разработаны классы и модули приложения (сколько, особенности);

Продумать и изложить перспективы[5] в усовершенствовании разрабатываемого приложения.

[5] Что хотелось бы реализовать, но не было сделано

Рекомендуемая литература:

1. Конспект лекций по курсу "Програмирование".

2. С.А.Рыбалка, Г.И.Шкатова. Методические указания «Языки программирования и методы трансляции» — Томск: изд. ТПУ, 2000 г. — 88 с.

3. Страуструп Б. Язык программирования C++. — М.: Радио и связь, 1991 г. — 352 с.