Государственное бюджетное профессиональное образовательное учреждение Ростовской Области
«Таганрогский колледж морского приборостроения»

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ К ВЫПОЛНЕНИЮ КУРСОВОГО ПРОЕКТА

по МДК 01.02 Прикладное программирование
ПМ 01 Разработка программных модулей программного обеспечения для компьютерных систем
для студентов специальности 09.02.07 «Программирование в компьютерных системах»

2021

ОДОБРЕНА цикловой комиссией
дисциплин программирования

УТВЕРЖДАЮ
Заместитель директора по УР

Протокол №____от _____ _____ О.Н. Морозова
Председатель ЦК «_____»_____
_____ О.В. Малыхина 2021 г.

Организация-разработчик:	Государственное бюджетное профессиональное образовательное учреждение Ростовской области «Таганрогский колледж морского приборостроения».
Разработчик:	О.В. Малыхина, преподаватель ГБПОУ РО «ТКМП».
Рецензенты:	Р.К. Пикара, генеральный директор ООО «ТактиМет»

Содержание

Введение

Цель курсового проекта: получить практические навыки в создании приложений с использованием высокоуровневых методов программирования

Порядок выполнения работы

1. Получить у преподавателя вариант задания.
2. Изучить методическое указание.
3. В папке для курсового проекта 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. Последним-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.
4. Предусмотреть ***постоянное копирование документа на другие носители***для уменьшения потерь в случае повреждения и удаления папок.
5. Распечатать пояснительную записку и сдать ее для проверки.
6. Защитить работу перед комиссией.

Требования

1. Исходные данные изначально следует разместить в файле данных.
2. В проекте должен быть предусмотрен отдельный модуль, в котором должен быть размещен созданный класс (АТД, компонент).
3. Изготавливаемый программный продукт должен быть хорошо прокомментирован, написан в соответствии с требованиями структурного программирования.
4. Оформление пояснительной записки следует выполнять согласно требованиям ГОСТ 2.105-2019.
5. Для защиты курсового проекта рекомендуется подготовить презентацию, в которую следует включить следующие слайды: постановка задачи, интерфейс класса, интерфейсы с результатами, блок-схемы наиболее сложных алгоритмов, выводы.

1 Методические указания

Как это следует из цели, поставленной в курсовом проекта, основная задача, стоящая перед студентом курса: научиться составлять программы высокого качества. Такие программы должны быть легко модифицируемыми, простыми в обращении. Они должны быть написаныс 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?

2 Структура и содержание курсового проекта

Титульный лист
Задание
Аннотация
Содержание
Введение

1. Общая часть
   1.1. Постановка задачи
   1.2. Анализ и исследование задачи, построение модели системы
   1.3. Разработка архитектуры решения
   1.4. Обоснование и выбор средств разработки решения
   1.5. Формализация расчетов
2. Описание программы
3. Тестирование программы
   3.1. Разработка плана тестирования.
   3.2. Оценка результатов проведения тестирования

Заключение
Список использованных источников

3 Содержательное наполнение разделов пояснительной записки

Аннотация

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

Введение

Введение должно содержать общие сведения по теме курсового проекта. 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 введениитакже 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 с применением современных технологий программирования);
- задачи[2], решаемые в проекте (это может быть построение математической модели[3], создание класса, использование экземпляров класса для принятия решений,…);
- используемые модели программирования (например, императивное программирование, модульное программирование, структурное программирование, объектно-ориентированное программирование, основанное на классах,…)
- практическую значимость полученных результатов (где можно использовать);
- какого рода ресурсы необходимы для реализации (ПК, программное обеспечение…уточнить какое);
- перспективы совершенствования изготавливаемого программного продукта.

1 Общая часть

1.1 Постановка задачи

Составляющие элементы этого разделам следующие:

- формулировка условия задачи;
- сборинформации 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.).

Формулировка условия задачи выдается преподавателем.

Сборинформации о задаче и выделение физического объекта. Следует определиться, с какими данными разработчик проекта имеет дело.При этом необходимо выявить самые существенные свойства, необходимые для решения задачи. Выделив наиболее важные факторы, можно пренебречь менее существенными.

---

[1] 2-3 предложения
[2] Уточнить каких решений в соответствии с темой
[3] Уточнить

1.2 Анализ и исследование задачи, построение модели

Составляющими данного раздела могут быть следующие элементы:

- выделение математического объекта;
- анализ существующих аналогов;
- разработкаматематической модели;

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

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 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.

1.3 Разработка архитектуры решения

Архитектура идентифицирует главные компоненты системы и способы их взаимодействия. Также это выбор таких решений, которые интерпретируются как основополагающие и не подлежащие изменению в будущем.

Составляющими данного раздела могут быть следующие элементы:

- выделение основных компонент программы на основе ранее рассмотренных аналогов
- разработка бизнес процессов функционирования
- разработка диаграмм, уточняющих функциональность

1.4 Обоснование и выбор средств разработки решения

Составляющими данного раздела могут быть следующие элементы:

- анализ возможных технических и программных средств.

1.5Формализация расчетов

Составляющими данного раздела могут быть следующие элементы:

– разработка моделей поведения математических объектов

– проектирование компонент программного решения

– разработка и описание блок-схем алгоритмов

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).

2Описание программы

Описание программы рекомендовано производить согласно ГОСТ 19.402-78. Пример описание модуля отображения данных профилографа.

2.1ОБЩИЕ СВЕДЕНИЯ

Программная компонента отображения данных профилографа представляет собой встраиваемый программных модуль, предоставляющий функциональность для отображения данных, вырабатываемых профилографом.
Модуль реализован на языке C++ и совместим со стандартом ANSIC++. При разработке данного модуля использовалась библиотека MFC.
Модуль предназначен для функционирования в среде Windows.

2.2 ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ

Программная компонента отображения данных профилографа предназначена для решения следующих задач:

- отображение данных, поступающих от профилографа;
- предоставление оператору органов управления системой отображения.

2.3 ОПИСАНИЕ ЛОГИЧЕСКОЙ СТРУКТУРЫ

Программная компонента отображения данных профилографа состоит из нескольких функциональных компонент:

- программная компонента системных функций;
- программная компонента отображения данных;
- программная компонента конфигурирования.

2.3.1 Программная компонента системных функций

Программная компонента системных функций представляет собой набор классов, реализующих следующие функции:

- обеспечение обмена сообщениями между неграфическими объектами;
- обеспечение выбора и автоматической смены текущей палитры отображения информации во всех графических объектах;
- набор функций для упрощенного создания базовых графических объектов;
- реализация очередей обмена данными;
- реализация базовой функциональности для графических объектов, используемых в программной компоненте отображения данных;
- организация таймера.

2.3.2 Обеспечение обмена сообщениями между неграфическими объектами

Обмен сообщениями между неграфическими объектами позволяет организовать отложенный вызов методов одного объекта другим.

Принцип организации обмена сообщениями такой же, как и принцип обмена сообщениями между окнами Windows.

В программе создается очередь сообщений, куда помещаются все отправляемые сообщения, и из которой, они последовательно выбираются, и передаются адресату. Для

передачи и приема сообщений создается теневое окно, очередь которого, и используется в качестве глобальной очереди сообщений. Для постановки сообщений в очередь, оно отправляется при помощи метода PostMessage созданному окну, используя механизм стандартных сообщений Windows. В этом стандартном сообщении передается информация о передаваемом сообщении (указатель на передаваемый объект) и информация об объекте, который должен обработать передаваемое сообщение (указатель на конкретный объект). Соответственно, при использовании такого подхода необходимо, чтобы объект, представляющий собой сообщение, создавался только динамически. Он будет удален автоматически после его обработки.

В компоненте системных функций определен интерфейс BaseEvent, который является базовым интерфейсом для всех объектов сообщений.

В данном интерфейсе определено перечисление EventsType. В данном перечислении содержится уникальный код для каждого, из используемых в системе сообщений. Для создания нового сообщения необходимо добавить в это перечисление новый уникальный номер.

Интерфейс BaseEvent имеет один конструктор, который на вход принимает номер типа сообщения. Стоит обратить внимание на то, что в данном интерфейсе запрещается конструктор копирования. Поэтому, если возникнет необходимость копировать сообщения, то необходимо будет определять непосредственно в каждом классе сообщений конструктор копирования.

В данном интерфейсе так же имеется один метод – check_event. Он определяет, является ли данное сообщение требуемого типа.

Интерфейс BaseEventHandle является базовым для всех объектов, которые могут обрабатывать сообщения. В нем определен один абстрактный метод – handle_event. Данный метод принимает указатель на класс BaseEvent. Прежде чем преобразовывать указатель от класса BaseEvent к требуемому классу сообщений, необходимо сначала проверить при помощи метода check_event, является ли это сообщение требуемым.

Вся работа с сообщениями организуется при помощи класса EventsQueue. Данный класс должен быть сначала создан, для того, чтобы механизм сообщений стал доступным. Класс EventsQueue имеет метод post_event, который оправляет сообщение заданному объекту. Первым параметром он принимает указатель на объект, который должен обработать сообщение, а вторым параметром – указатель на передаваемое сообщение.

Следует отметить, что в независимости от того, в каком потоке была инициирована передача сообщения, обработка сообщения будет производиться всегда в главном потоке программы.

2.3.3 Обеспечение выбора и автоматической смены текущей палитры отображения информации во всех графических объектах

Обеспечение выбора и автоматической смены текущей палитры отображения информации во всех графических объектах позволяет программисту очень легко реализовать возможность отображения графической информации в рамках одного приложения с использованием единой палитры. При этом, при смене текущей палитры происходит автоматическая ее замена во всех элементах отображения информации, что снимает с программиста необходимость при смене палитры вручную заботится, о том, чтобы каждый из элементов управления сменил палитру отображения.

Данная функциональность состоит из трех основных блоков: палитра, сервер палитр, клиент.

Класс палитры обеспечивает преобразование числового значения, поступающего ему на вход, в код цвета, соответствующий этому числовому значению в текущей палитре.

В библиотеке системных функций определен базовый интерфейс для всех палитр – Palette. Каждая палитра идентифицируется текстовым дескриптором. В данном классе определены два интерфейса:

- virtualintget_colors_count() = 0 – данный метод возвращает число цветов в данной цветовой палитре;
- virtualintget_color(intindex) = 0 – данный метод возвращает цветовое значение, соответствующее указанному индексу (Максимальное значение индекса get_colors_count() – 1).

Так же в данном классе имеется несколько дополнительных методов:

- virtualboolcheck_descriptor(constCString&check_descriptor) – данный метод возвращает true, если переданный текстовый дескриптор совпадает с дескриптором палитры;
- virtualvoidfill_descriptors_list(CList<CString, CString>* list) – данный метод добавляет свой дескриптор в список дескрипторов палитр.

В данной программной компоненте представлены три палитры: стандартная сине-зелено-красная (NotmalPalette), монохромная зеленая (GreenPalette), коричневая (BrownPalette).
Сервер палитр реализует механизм хранения и выбора палитр.
Сервер палитр реализует класс PalettesContainer.
Он имеет следующие методы:

- staticvoidadd_pallete(Palette* palette) – данный метод добавляет на сервер новую палитру;
- staticPalette* get_current_palette() – данный метод возвращает указатель на текущую палитру;
- staticPalette* get_palette(constCString&descriptor) – данный метод возвращает указатель на палитру, по её текстовому дескриптору;
- staticvoidset_current_palette(constCString&descriptor) – данный метод устанавливает текущую палитру по ее текстовому дескриптору;
- staticvoidadd_palettes_owner(BaseEventHandler* object) – данный метод регистрирует объект, который будет получать уведомление о смене палитры (событие BaseEvent::PaletteChange);
- staticvoidremove_palettes_owner(BaseEventHandler* object) – данный метод удаляет объект из списка рассылки уведомлений;
- staticboolcheck_current_palette(constCString&descriptor) – данный метод проверяет, является ли запрашиваемая палитра текущей;
- staticCList<CString, CString>* get_palettes_list() – данный метод возвращает список дескрипторов всех доступных палитр. После использования данный список должен быть удален.

При смене палитры класс PalettesContainer посылает всем объектам, зарегистрированным при помощи метода (add_palettes_owner), событие PaletteChangeEvent, в котором содержится установленная палитра. При получении данного сообщения, все классы должны заменить у себя используемую палитру на вновь установленную.
Для упрощения работы с палитрами, в программной компоненте реализован класс (Colors), преобразующий значения из заданного числового диапазона, в значения цвета.
Класс Colors имеет следующие методы:

- Colors(float range, Palette* palette = 0) – конструкторкласса. Принимает на вход значение числового диапазона, для которого будет осуществляться преобразование. Вторым параметром в класс может быть передана конкретная палитра, в этом случае он не будет осуществлять автоматически смену текущей палитры;
- voidset_range(floatvalue) – данный метод устанавливает новый диапазон значений;

- intget_color(floatvalue) – данный метод возвращает цвет, для переданного значения;
- voidupdate_palette() – данный метод заставляет класс принудительно сменить текущую палитру.

При создании объект класса Colors, в случае если второй параметр конструктора был равен 0, регистрирует себя на сервере палитр PlattesContainer. После чего, он будет автоматически обрабатывать все уведомления от сервера палитр, и реагировать на изменение текущей палитры.

Если при создании объекта класса Colors, в качестве второго параметра была указана конкретная палитра, то класс не будет реагировать на изменение текущей палитры. Для этого необходимо вызвать метод update_palette, после выбора новой текущей палитры.

2.3.4 Набор функций для упрощенного создания базовых графических объектов

При работе с базовыми графическими объектами (окна и битовые образы) с использованием функций WinAPI, приходится вызывать набор функций и заполнять большие объемы управляющих структур данных, как для создания, так и для удаления данного вида объектов. Чтобы упростить программисту процесс создания и удаления базовых графических объектов, был разработан ряд функций, которые позволяли бы выполнять такие операции как, создание объектов, их удаление, путем вызова всего одного метода.

Такие функции инкапсулированы в классе BaseWindowLibrary. Следует отметить, что для всех методов данного класса введено описание принимаемого параметра. Если параметр является для метода входным, то перед его определением указывается ключевое слово IN, если параметр, передаваемый в метод принимает возвращаемое значение, то перед его определением указывается ключевое слово OUT.

Класс BaseWindowLibrary имеет следующие методы:

- static void create_bitmap(IN HWND window, OUT HDC *dc_bitmap, OUT HBITMAP *bitmap). Данный метод создает bitmap для указанного в параметре window окна. Размер создаваемого битового образа будет равен размеру окна. Метод возвращает в переменной dc_bitmap контекст устройства (devicecontext) для создаваемого битового образа. В переменной bitmap возвращается дескриптор созданного битового образа;
- static void create_bitmap(IN HWND window, IN constCRect&rect, OUT HDC* dc_bitmap, OUT HBITMAP* bitmap). Данный метод работает, как и предыдущий, за исключением того, что в переменной rect указывается размер создаваемого битового образа;
- static HPEN add_pen_to_bitmap(IN HDC dc_bitmap, IN int style, IN int width, IN COLORREF color). Данный метод добавляет к созданному битовому образу объект pen. В параметре dc_bitmap передается контекст устройства битового образа, в параметре style указывается стиль, которым на битовом образе будут отображаться линии, width – указывает толщину рисования линий, color – указывает цвет рисования линий.
- static void delete_bitmap(IN HDC dc_bitmap, IN HBITMAP bitmap). Данный метод удаляет ранее созданный битовый образ. Метод принимает в качестве параметра dc_bitmap контекст устройства битового образа, в качестве параметра bitmap метод принимает дескриптор созданного битового образа;
- static HWND create_window(IN HWND parent_window, IN const char* window_name, IN WNDPROC window_proc). Данный метод создает дочернее окно. Метод принимает следующие входные параметры: parent_window – дескриптор родительского окна, window_name – имя

создаваемого окна (не может быть пустым), window_proc – указатель на метод обработки сообщений от создаваемого окна. В качестве возвращаемого значения метод возвращает дескриптор созданного окна. Созданное окно имеет размер, равный размеру клиентской зоны родительского окна;

- static HWND create_window(IN HWND parent_window, IN constCRect&rect, IN const char* window_name, IN WNDPROC window_proc, IN int style = WS_CHILD). Данный метод работает так же, как и предыдущий. Но принимает ряд дополнительных параметров: rect – размеры создаваемого окна, style – стиль создаваемого окна.

2.3.5 Очереди обмена данными

Очереди обмена данными служат для организации буферизированного обмена между частями программы. В программной компоненте системных функций реализована только одна очередь, организованная по принципу FIFO.

Она реализована в виде шаблона FIFOQueue<classT>, где T – это тип данных, которым будет производиться обмен.

Данный шаблон имеет следующие методы:

- voidappend(T* data). Данный метод добавляет блок данных в очередь;
- T* get_next(). Данный метод удаляет блок данных из очереди, возвращая указатель на него;
- boolempty(). Данный метод возвращает true, если очередь пустая.

2.3.6 Базовая функциональность графических объектов

В разрабатываемой программной компоненте отображения данных профилографа имеется ряд графических объектов, которые должны реагировать на ряд событий (изменение размеров, необходимость обновления своего содержимого, необходимость отображения графической информации на экран). При этом, их реализация на уровне WinAPI носит схожий характер.

Для обеспечения унификации и облегчения разработки таких объектов, в программной компоненте системных функций была реализована базовая функциональность для таких графических объектов в виде класса ProfilographBaseScale.

Данный класс имеет следующие методы:

- ProfilographBaseScale(HWND parent_window). Этоконструкторобъекта. В качестве входного параметра он принимает дескриптор родительского окна.
- voidresize(). Данный метод должен быть вызван, когда родительское окно меняет свои размеры;
- voidfill_scale(). Вызов данного метода приведет к тому, что объект обновит информацию в своем внутреннем графическом буфере;
- voidapplay_changes(). Данный метод должен быть вызван, после изменения настроек программы.
- voidrepaint(). Вызов данного метода приведет к обновлению изображения на экране.

Также в данном классе определены два интерфейса, которые необходимо перегружать в дочерних объектах:

- virtual void fill_bitmap(HDC dc, HBITMAP bitmap, constCRect&rect) = 0. Данный метод вызывается, когда необходимо заполнить bitmap базовым

изображением (например, нанести сетку), в том числе и проинициализировать bitmap (например, рассчитать коэффициенты отображения). В качестве параметров в метод передаются контекст устройства для bitmap, дескриптор bitmap и геометрические размеры bitmap.;

- virtual void paint_bitmap(HDC dc, HBITMAP bitmap, constCRect&rect). Данный метод вызывается, когда необходимо отобразить на bitmap регулярную информацию. Параметры такие же, как для предыдущего метода.

Также в этом классе определены методы доступные только для дочерних объектов:

- voidupdate_bitmap(). Вызов этого метода приведет к вызову метода paint_bitmap, а затем отображению bitmap на экран.

2.3.7 Таймер

В программной компоненте системных функций реализован таймер, который посылает уведомление в виде объектов событий.

Таймер реализуется классом Timer, который имеет следующие методы:

- Timer(BaseEventHandler* parent, int period, int ID = -1). Конструктор класса. Принимает на вход следующие параметры: родитель (объект, которому будут посылаться уведомления, об истечении заданного периода времени), значение периода таймера, и необязательный параметр – числовой идентификатор таймера;
- voidstart(). Запускает таймер. После истечения заданного периода, родителю посылается событие TimerEvent, содержащее числовой идентификатор таймера, затем таймер автоматически перезапускается;
- voidstop(). Данный метод останавливает таймер;
- boolcheck_ID(intID). Данный метод проверяет идентификатор таймера.

2.4 Программная компонента конфигурирования

Программная компонента конфигурирования предоставляет оператору средства для настройки программной компоненты отображения данных и отображению навигационной информации.

Программная компонента конфигурирования состоит из нескольких частей:

- блок параметров;
- блок выбора настоек;
- панель информации.

2.4.1 Блок параметров

Блок параметров представляет собой набор переменных, значения которых управляют процессом отображения данных профилографа. Набор параметров реализован в виде static переменных класса ProfilographParameters. В данном классе определены следующие параметры:

- staticScaleTypescale_type. Тип горизонтальной шкалы (метры или секунды). Задается из диалога конфигурирования пользователем;

- staticintdistance_scale_step. Размер шага сетки по оси X в пикселях. Задается из диалога конфигурирования пользователем;
- staticfloatdistance_scale. Определяет значение величины метр/пиксель. Рассчитывается автоматически при изменении соответствующих настроек;
- staticfloattime_scale. Определяет значение величины время/пиксель. Рассчитывается автоматически при изменении соответствующих настроек;
- staticboolscale_draw. Определяет, надо ли отображать сетку. Задается из диалога конфигурирования пользователем;
- staticbooldepth_scale_draw. Определяет, надо ли отображать сетку глубин. Задается из диалога конфигурирования пользователем;
- staticboolmeters. Определяет единицы измерения шкалы.
- staticintmax_depth. В данной переменной хранится максимальное отображаемое значение глубины;
- staticbooldraw_marking. В данной переменной хранится режим отображения маркеров (отображать/не отображать);
- staticintmarking_step. В данной переменной задается шаг отображения маркеров в пикселях;
- staticintcurrent_law. В данной переменной хранится текущий закон отображения данных.
- staticfloatFD. В данной переменной храниться текущая частота дискретизации;
- staticfloatsound_speed. В данной переменной храниться текущая скорость звука;
- staticfloatspeed. В данной переменной хранится текущая скорость корабля носителя;
- staticboolshow_amplitude. Данная переменная определяет, будут ли отображаться амплитудные отметки;
- staticCStringimages_dir. В данной переменной хранится имя директории, в которую будут сохраняться копии экрана;
- staticintperiod. В этой переменной хранится период сохранения экрана в миллисекундах;
- staticboolsave_images. Эта переменная определяет, будет ли производиться периодическое сохранение экрана.

2.4.2 Блок выбора настроек

Блок выбора настроек реализуется классом ProfilographControl. Данный класс реализует диалоговое окно со стандартными элементами ввода, хранимое в файле ресурсов (IDD_PROFILOGRAPH_CONTROL).
Данное диалоговое окно имеет следующие элементы ввода:

- IDC_RADIO_TIME. Если выбран этот элемент, то отображение данных будет производиться по временной шкале;
- IDC_RADIO_DISTANCE. Если выбран этот элемент, то отображение данных будет производиться по шкале дистанций;
- IDC_RADIO_AMPLITUDE_ON. Если выбран этот элемент, то будет производиться отображение амплитудных отметок;
- IDC_RADIO_AMPLITUDE_OFF. Если выбран этот элемент управления, то не будет производиться отображения амплитудных отметок;
- IDC_STATIC_PALETTES_PARENT_WINDOW. Родительское окно для ComboBox, производящего выбор текущей палитры;

- IDC_RADIO_SCALE_ON. Если выбран этот элемент, то будет производиться отображение сетки;
- IDC_RADIO_SCALE_OFF. Если выбран этот элемент, то не будет производиться отборажение сетки;
- IDC_COMBO_DISTANCE_STEP. Данный всплывающий список задает шаг отображение шкалы, для шкалы дистанций;
- IDC_COMBO_DISTANCE_STEP. Данный всплывающий список задает шаг отображения шкалы, для шкалы времени;
- IDC_RADIO_MARKS_ON. Если выбран этот элемент, то будет производиться отображение маркеров;
- IDC_RADIO_MARKS_OFF. Если выбран этот элемент, то отображение маркеров производиться не будет;
- IDC_COMBO_DISTANCE_COFF. Данный всплывающий список позволяет задать шаг отображения маркеров по шкале дистанций;
- IDC_COMBO_TIME_COFF. Данный всплывающий список позволяет задать шаг отображения маркеров по оси времени;
- IDC_RADIO_DEPTH_SCALE_ON. Если выбран данный элемент, то будет производиться отображение сетки глубин;
- IDC_RADIO_DEPTH_SCALE_ON. Если выбран данный элемент, то не будет производиться отображение сетки глубин;
- IDC_COMBO_DEPTH. Данный всплывающий список позволяет выбирать максимальную отображаемую глубину;
- IDC_COMBO_LAW. Данный всплывающий список позволяет выбирать номер закона отображения данных;
- IDC_STATIC_LAW. Родительское окно для отображения текущего закона отображения;
- IDC_EDIT_FOLDER_NAME. Данное поле ввода позволяет задать папку, в которую будет сохраняться текущее изображение экрана;
- IDC_EDIT_PERIOD. Данное поле ввода позволяет задать период сохранения текущего экрана.

Выбор палитры для отображения производится при помощи класса PalettesComboBox. Данный класс определяет доступные палитры в системе, и предлагает пользователю осуществить выбор одной из доступных палитр, представляя данные в графическом виде.

Данный класс имеет следующие методы:

- PalettesComboBox(CWnd* parent, constCRect&rect, int ID = -1). Конструктор класса. Он принимает следующие параметры: родительское окно, на котором будет расположен ComboBox, размеры зоны редактирования для ComboBox, и в качестве необязательного параметра принимается числовой идентификатор объекта. По этому числовому идентификатору через таблицу сообщений можно получать уведомления об изменении состояния ComboBox;
- voidapplay_changes(). Вызов данного метода устанавливает в системе текущей палитру, выбранную в данный момент в ComboBox.

При выборе закона отображения полученная палитра отображения данных отображается классом ProfilogrpahDrawLaw. Данный класс имеет следующие методы:

- ProfilographLawDraw(HWND parent_window). Конструкторкласса. В качестве параметра принимает дескриптор родительского окна. Отображение законов будет производиться на всю площадь клиентской зоны родительского окна;
- voidset_law(intlaw). Данный метод устанавливает номер текущего закона отображения;

- voidupdate_palette(). Данный метод заставляет объект установить текущую палитру системы.

При открытии диалогового окна из метода DoDataExchange вызывается метод load_values, в котором все элементы ввода принимают значения в соответствии со значениями, хранимыми в классе ProfilographParameters.

Если при закрытии окна была нажата кнопка «OK», то автоматически вызывается метод applay_values, в котором все элементы ввода заносят вновь установленные значения в переменные класса ProfilographParameters, после чего родителю посылается сообщение ApplaySettingsEvent.

Внешний вид окна «Настройки» представлен на рисунке 3.2.

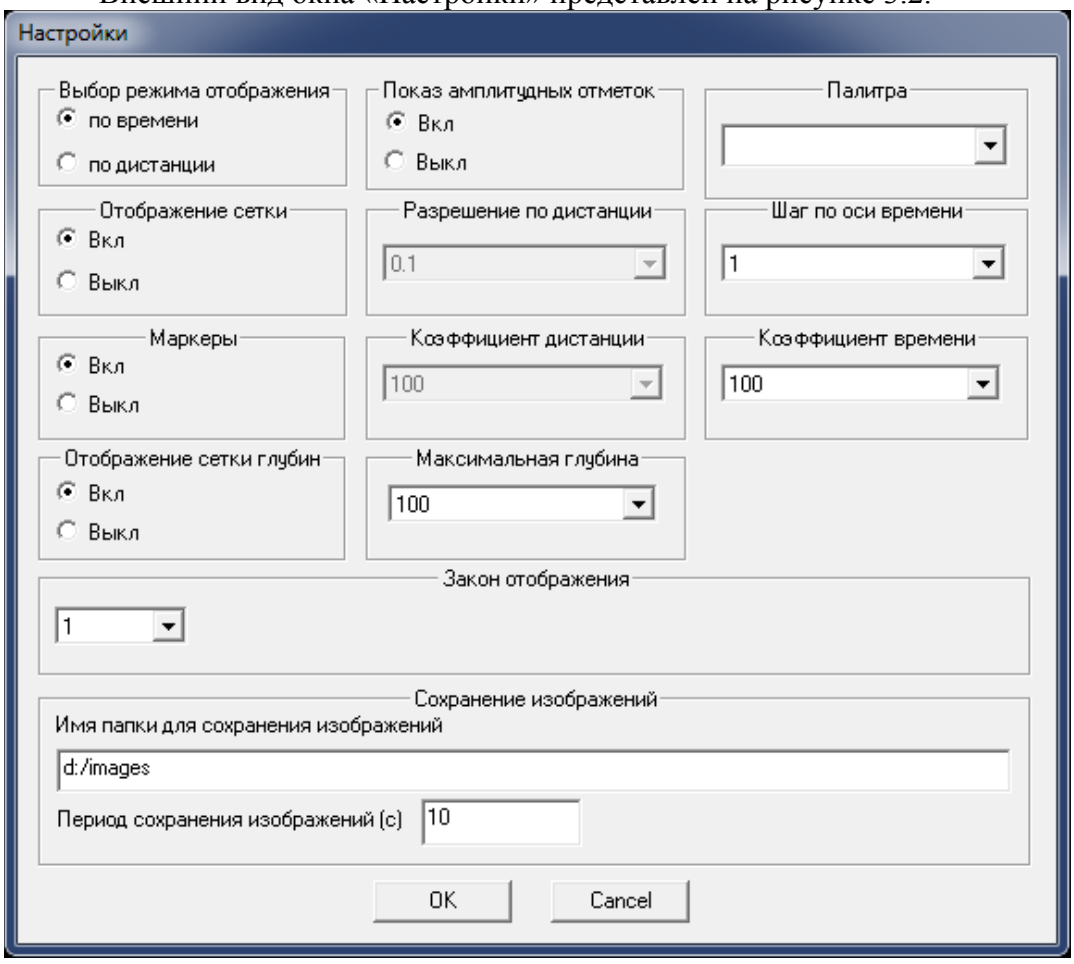


Рисунок 3.2- Внешний вид окна «Настройки»

2.4.3 Панель информации

Панель информации служит для отображения навигационной информации, предоставления оператору элементов вызова окна настроек и элементов сохранения текущего экрана.

Панель информации реализуется классом ProfilographInfo. Данный класс является диалоговым окном, считываемым из файла ресурсов (IDD_FORMVIEW2), и имеет следующие элементы:

- IDC_STATIC_DEPTH. Родительское окно для отображения текущей глубины;
- IDC_COURSE. Элемент отображения курса корабля;
- IDC_SHIP_PATH. Элемент для отображения пройденного пути корабля;
- IDC_KREN. Элемент для отображения крена корабля;

- IDC_DIFFERENT. Элемент для отображения дифферента корабля;
- IDC_SHIP_SPEED. Элемент для отображения скорости корабля;
- IDC_SOUND_SPEED. Элемент для отображения текущей скорости звука;
- IDC_STATIC_LONGITUDE. Элемент для отображения текущей долготы;
- IDC_STATIC_LATITUDE. Элемент для отображения текущей широты;
- IDC_STATIC_TIME. Элемент для отображения текущего времени;
- IDC_STATIC_DATE. Элемент для отображения текущей даты;
- IDC_BUTTON1. Копка вызова диалога настроек;
- IDC_BUTTON_SAVE_SCREEN. Кнопка включения/выключения сохранения текущего экрана.
- IDC_BUTTON_KADR. Кнопка мгновенного сохранения экрана. Посылает родителю событие BaseEvent::ScreenSave.

Класс ProfilographInfo имеет следующие методы:

- ProfilographInfo(BaseEventHandler* parent_object, CWnd* parent_window = NULL). Конструктор класса. Принимает следующие параметры: родитель (объект, который будет получать уведомления от данного класса и от класса ProfilographControl);
- voiddepth_receive(floatdata). Данный метод устанавливает текущее значение глубины;
- voidnavigation_receive(constSNav&navigation). Данный метод устанавливает текущие навигационные данные.

Панель вывода навигационной информации представлена на рисунке 3.3.

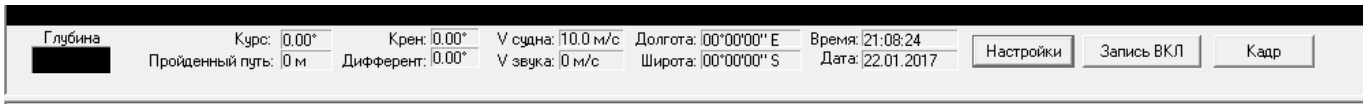


Рисунок 3.3 – Панель вывода навигационной информации

2.5 Программная компонента отображения данных

Программная компонента отображения данных является основной частью программной компоненты отображения данных профилографа. Она выполняет прием обработанных данных и отображение их в виде, заданным пользователем. Данная компонента реализуется классом ProfilographMainObject. Данный класс можно разделить на два блока: блок приема данных и блок отображения данных.

2.5.1 Блок приема данных

Для приема данных организуются три очереди данных, организованных на основе классов DataOutQueue (две очереди) и DoubleDataQueue(одна очередь). При помощи данных очередей передаются данные непосредственно профилографа, данные о линии дна, амплитудные отметки эхолота.

Прием всех данных осуществляется следующим образом:

- для приема данных по очередям DataOutQueue заводится по одному параллельному потоку приема данных;
- в приемной части организуются два циклических буфера на основе класса FIFOQueue;
- в каждом потоке происходит ожидание события от очереди приема данных;
- после того, как событие становится активным, что свидетельствует о появлении данных в очереди, поток приема данных читает последовательно все данные из очереди данных, записывая их в циклический буфер;

- после того, как все данные прочитаны, основной программе посылается событие DepthDataReceiveEvent (если были приняты данные о текущей глубине) или событие ProfilographDataReceiveEvent (если были приняты данные профилографа);
- основная программа, получив одно из событий, проверяет, имеются ли данные, как о линии дна, так и для профилографа;
- если данные есть для линии дна и профилографа, то производится отображение данных, иначе ничего не происходит.

Прием данных осуществляется классом ProfilographMainObject. Прием данных о линии дна осуществляется потоком thread_procedure. Циклическая очередь для данных о линии дна – data_list. Прием данных профилографа осуществляется потоком profilograph_thread_procedure. Циклическая очередь для данных профилографа – profilograph_data_list.

Все принимаемые данные сохраняются в промежуточном циклическом буфере ProfilographDataContainer, на случай если необходимо обработать заново часть последних данных (например, при изменении текущей максимальной отображаемой глубины).

Данный класс создается в классе ProfilographMainObject, и имеет следующие методы:

- structItem. Тип хранения данных. В данной структуре хранятся данные эхолота и профилографа;
- const Item& operator[](int index) const. Возвращает блок данных по смещению, относительно последнего сохраненного блока;
- void append(constFathometerPeriodData&append_data, constOnLineTestPeriodData&append_profilograph_data). Добавляет на хранение очередной блок данных. Если места в буфере нет, то перетирается самый последний сохраненный блок данных;
- intcount() const. Возвращает число реально хранящихся блоков данных.

2.5.2 Блок отображения данных

Блок отображения данных класса ProfilographMainObject реализует следующие основные задачи:

- отображение вертикальной и горизонтальной шкал;
- отображение поля маркеров;
- отображение непосредственно самих данных.

Блок отображения данных создает для отображения две горизонтальные шкалы: шкалу времени и шкалу дистанций. Одновременно отображается только одна из них (в зависимости от значения переменной ProfilographParameters::scale_type).

Шкала дистанций реализуется классом ProfilographDistanceScale. Данный класс наследуется от класса ProfilographBaseScale. Из доступных методов данный класс имеет только конструктор, который на вход принимает только один параметр – дескриптор родительского окна. Шкала будет автоматически занимать всю клиентскую область родительского окна.

Шкала времени реализуется классом ProfilographTimeScale. Данный класс наследуется от класса ProfilographBaseScale. Он имеет следующие методы:

- ProfilographTimeScale(HWND parent_window, intoffset_from_border). Конструктор класса. Первым параметром принимает дескриптор родительского окна, вторым - величину смещения от правого края экрана;
- voidcorrect_scale(). Данный метод заставляет передвинуть шкалу времени на один пиксель. Он должен вызываться при каждом приеме данных.

Вертикальная шкала (шкала глубин) реализуется классом ProfilographDepthScale. Данный класс наследуется от класса ProfilographBaseScale. В классе дополнительно определен только конструктор, принимающий на вход дескриптор родительского окна.
Отображением линий шкал на отображаемых данных занимается класс ProfilographScaleDraw. Данный класс имеет следующие методы:

- ProfilographScaleDraw(HWND parent_window). Конструкторкласса. Входной параметр – дескриптор родительского окна;
- voiddraw_scale(HDCdc). Данный метод отображает линии шкал (в соответствии с текущими настройками) на указанный контекст устройства;
- voidcorrect_scale(). Данный метод сдвигает вертикальные линии на один пиксель влево;
- voidresize(). Данный метод должен вызываться, когда меняются геометрические размеры родительского окна;
- voidapplay_settings(). Данный метод должен вызываться, когда меняются настройки программы.

Линии маркеров глубин на отображаемых данных рисует класс ProfilographDepthsMark. Данный класс наследуется от класса ProfilographBaseScale. Класс ProfilographDepthsMark имеет следующие методы:

- ProfilographDepthDraw(HWND parent_window). Конструкторкласса. Входной параметр – дескриптор родительского окна;
- voiddepth_receive(floatdata). Данный метод вызывается, когда получены новые данные о глубине.

Отображением данных занимается класс ProfilographDraw. Данный класс принимает значение глубины и данные профилографа, производит их масштабирование в соответствии с текущими настройками, и отображает их.
Данный класс имеет следующие методы:

- ProfilographDraw(HWND parent_window, BaseEventHandler* parent). Конструктор класса. Принимает в качестве параметров дескриптор окна и указатель на объект родителя (для будущего использования);
- voidresize(). Данный метод должен вызываться, когда родительское окно меняет свои размеры;
- intdepth_receive(constFathometerPeriodData& data, const Real32Vector&profilograph_data). Данный метод вызывается, когда поступают новые данные. Первый параметр – текущая глубина, второй – массив значений данных профилографа. Возвращаемое значение – число столбов отображенных данных (для шкалы времени всегда 1);
- voidupdate(). Вызов данного метод приведет к перерисовке изображения на экране;
- voidapplay_settings(). Данный метод должен вызываться, когда меняются настройки программы.

Подготовленные данные классом ProfilographDraw отображаются классом ProfilographBitmapDraw. Он имеет следующие методы:

- ProfilographBitmapDraw(HWND parent_window). Конструкторкласса. Принимает в качестве параметра дескриптор окна родителя;
- voidresize(). Данный метод должен вызываться, когда родительское окно меняет свои размеры;
- draw_data(int x, conststd::vector<int>& data, intdata_count, int begin). Данный метод вызывается, когда поступают новые данные. Первый параметр – текущая глубина, второй – массив значений данных профилографа, третий – количество данных, четвертый – первый пиксель, с которого надо производить отображение данных;

- voiddraw_bitmap(HDCdc). Данный метод отображает данные на указанный контекст устройства;
- voidapplay_settings(). Данный метод должен вызываться, когда меняются настройки программы.

Амплитудные отметки отображаются классом ProfilographAmplitudeDraw. Он имеет следующие методы:

- ProfilographAmplitudeDraw(HWND parent_window). Конструкторкласса. Принимает в качестве параметра дескриптор окна родителя;
- voidset_data(float depth, const Real32Vector& data). Отображает амплитуду сигнала. Первый параметр – текущая глубина (не используется), второй – массив отображаемых данных;
- voidset_data(float depth, const Int16Vector& data). Тоже, что и предыдущий метод, отличие в формате принимаемых данных.

Пример окна отображения данных представлен на рисунке 3.4

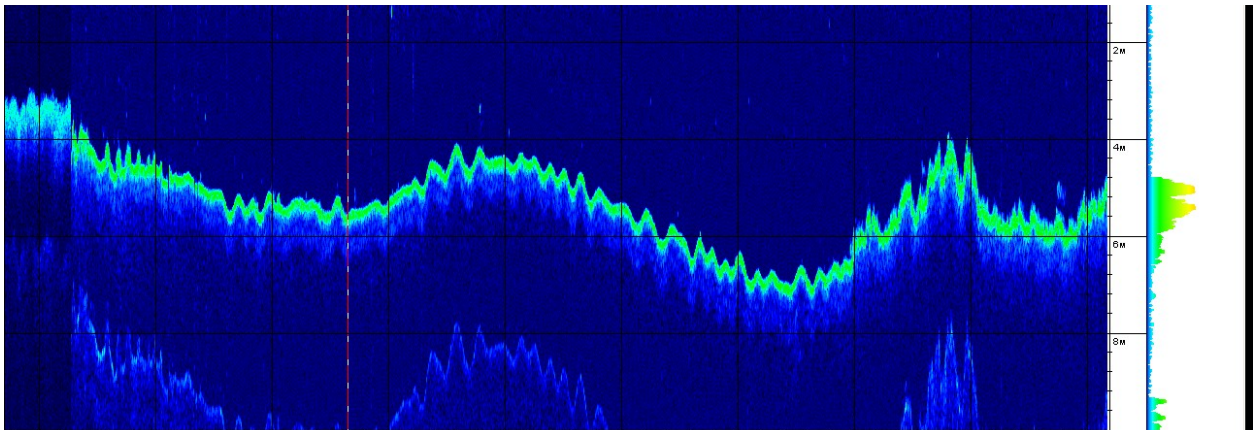


Рисунок 3.4

2.6 ИСПОЛЬЗУЕМЫЕ ТЕХНИЧЕСКИЕ СРЕДСТВА

Программная компонента отображения данных профилографа предназначения для функционирования на платформе «Intel» под управлением ОС Windows.

2.7 ВЫЗОВ И ЗАГРУЗКА

Программная компонента отображения данных профилографа представляет собой встраиваемый модуль. Для ее создания необходимо создать объект класса ProfilographMainObject. В качестве параметра parent_window – указывается дескриптор родительского окна, data_queue – указатель на очередь с данными о глубине, profilograph_data_queue – указатель на очередь с данными профилографа, amplitude_queue – очередь с данными об амплитудных отметках.
Объект через очереди будет принимать и отображать данные профилографа.
При изменении размеров родительского окна необходимо вызывать метод resize.
При изменении настроек программы необходимо вызывать метод applay_settings.

2.8 ВХОДНЫЕ ДАННЫЕ

Программная компонента отображения данных профилографа принимает на вход следующие данные:

- FathometerPeriodData – данные о текущей глубине;
- OnLineTestPeriodData – данные профилографа;
- Int16Vector – данные амплитудных отметок.

2.9ВЫХОДНЫЕ ДАННЫЕ

Программная компонента отображения данных профилографа не имеет выходных данных.

3 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

Здесь 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);
6. …

Продумать и изложить перспективы[4] в усовершенствовании разрабатываемого приложения.

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

4 Требования к оформлению курсового проекта

4.1Оформление пояснительной записки

4.1.1Общие требования

Пояснительная записка выполняется с помощью компьютера на одной стороне листа белой бумаги формата А4 (размер листа 297х210 мм) с основной надписью расположенной вдоль короткой стороны листа, с соблюдением следующих требований:

-расстояние от рамки формы до границ текста в начале и конце строк не менее 5 мм;

-расстояние от верхней строки текста до верхней рамки - не менее 10 мм;

-расстояние от нижней строки текста до нижней рамки - не менее 8 мм;

4.1.2 Оформление основных надписей

Обязательным требованием при оформлении пояснительной записки курсового проекта является нанесение основной надписи.

Различают:

-основная надпись для первого листа графического документа (рисунок 4.1);

- основная надпись для первого листа текстового документа (рисунок ***4.2);***

- основная надпись для всех последующих листов текстовых и документов (рисунок 4.3).

Рисунок 4.1 – Основная надпись для чертежей и схем

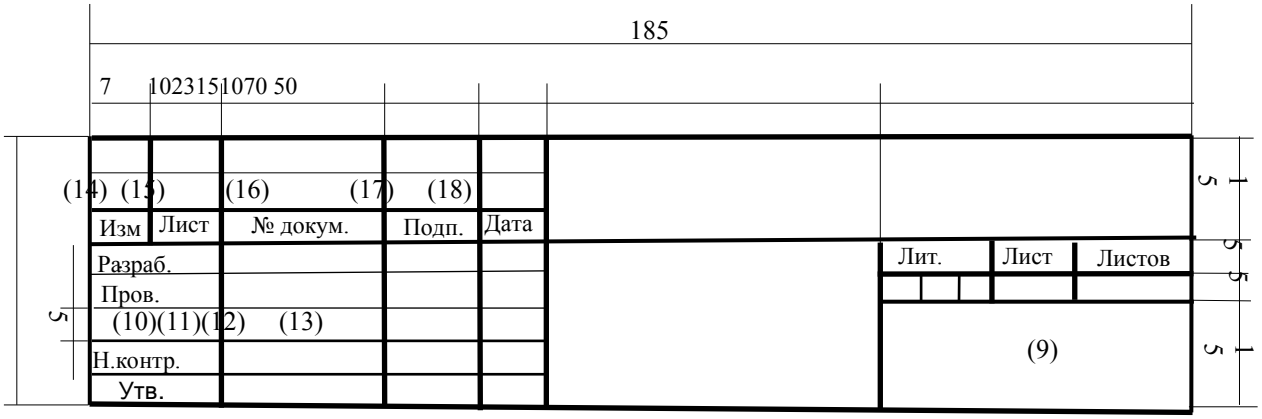


Рисунок 4.2 – Основная надпись для текстовых конструкторских документов (первый или заглавный лист)

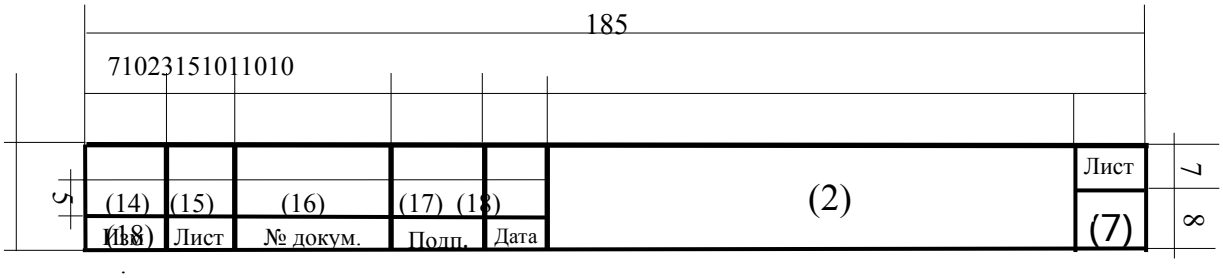


Рисунок 4.3 – Основная надпись для чертежей, схем и текстовых конструкторских документов (последующие листы)

Обязательным является заполнение следующих граф основной надписи (номера граф на рисунках 4.1 - 4.3 показаны в скобках):

-в графе 1 - указывается наименование изделия (в соответствии с требованиями ГОСТ 2.109-73), а также наименование документа;

- в графе 2 - обозначение документа (см. главу 3);

-в графе 9 - наименование предприятия, выпускающего документ (СГАЭК).

Подписи учащихся, разработавших данный документ, и лица, ответственного за проверну, являются обязательными.

4.1.3 Требования к оформлению текста

Поля

При оформлении:

- левое-30 мм;
- правое - не менее 15 мм;
- верхнее и нижнее - не менее 20 мм.

Выравнивание - по ширине.

Шрифт

Текст пояснительной записки выполняется шрифтом TimesNewRoman черного цвета с высотой 14 пт.

Межстрочный интервал

Текст пояснительной записки выполняется с межстрочным интервалом равный множителю 1,5.

Абзацный отступ

Абзацы в тексте начинают отступом 15 мм, одинаковым по всему тексту.

Внесение коррекции в текст пояснительной записки

Допускается вписывать в отпечатанный текст отдельные слова, формулы, условные знаки, а также выполнять иллюстрации черными чернилами (пастой, тушью).

4.1.4 Нумерация разделов

Каждый раздел должен начинаться с нового листа. Разделы, подразделы следует нумеровать арабскими цифрами без точки в конце. Разделы должны иметь порядковый номер (1, 2 и т.д.). Нумерация подразделов включает номер раздела и порядковый номер подраздела, входящего в данный раздел, разделенные точкой (2.1, 2.2, 3.1 и т.д.).

Если документ не имеет подразделов, то нумерация пунктов в нем должна быть в пределах раздела, и номер пункта должен состоять из номеров раздела и пункта, разделенных точкой. В конце номера пункта точка не ставится. Ниже представлен пример оформления нумерации разделов и подразделов пояснительной записки:

1 Общая характеристика работы
1.1
1.2 } Нумерация пунктов первого раздела
1.3
2 Специальная часть

2.1
2.2 } Нумерация пунктов второго раздела

Если документ имее подразделы, то нумерация пунктов должна быть в пределах подраздела и номер пункта должен состоять из номеров раздела, подраздела и пункта, разделенных точками, например:

3 Методы испытаний
3.1 Планирование и результаты эксприментов
3.1.1
3.1.2 } Нумерация пунктов первого подраздела третьего раздела
3.1.3
3.2 Разработка многофакторных пользовательских регрессий
3.2.1
3.2.2 } Нумерация пунктов второго подраздела третьего раздела
3.2.3

Если раздел или подраздел состоит только из одного пункта, он также нумеруется.

Пункты, при необходимости, могут быть разбиты на подпункты, которые *должны иметь порядковую нумерацию* в пределах каждого пункта, например: 4.2.1.1, 4.2.1.2, 4.2.1.3 и т.д.

Внутри пунктов и подпунктов могут использоваться перечисления (списки). Перед каждой позицией перечисления следует ставить дефис или при необходимости ссылки в тексте документа на один или несколько элементов перечисления - строчную букву, после которой ставится скобка.

Для дальнейшей детализации перечислений необходимо использовать арабские цифры, после которых ставится скобка, а запись производится с абзацного отступа, как показано в примере:

а)_____
б)_____
1)_____
2)_____
в)_____

4.1.5 Оформление заголовков

Текст записки разбивается на разделы и подразделы. Разделы и подразделы должны иметь заголовки, которые выделяются полужирным шрифтом.

Заголовки разделов печатают прописными буквами без точки в конце, не подчеркивая, с выравниванием по центру по ширине основного текста без абзацного отступа.

Заголовки подразделов печатают с прописной буквы без точки в конце, не подчеркивая, с выравниванием по левому краю основного текста с абзацным отступом.

Переносы слов в заголовках не допускаются. Если заголовок состоит из двух предложений, их разделяют точкой.

Параметры форматирования заголовков разделов

Все заголовки разделов, а также структурных частей работы «СОДЕРЖАНИЕ», «ВВЕДЕНИЕ», «ЗАКЛЮЧЕНИЕ», «СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ», «ПРИЛОЖЕНИЯ» должны быть оформлены следующим образом:

шрифт TimesNewRoman, размер 14, все символы прописные;
выравнивание по центру, без абзацного отступа;
межстрочный интервал одинарный;
интервал перед — 0 пт;
интервал после — 6 пт;
положение на странице — с новой страницы.

Переносы слов в заголовках не допускаются. Точку в конце заголовка не ставят.

Параметры форматирования заголовков подразделов

Заголовки подразделов записывают *с абзаца строчными буквами* (кроме первой прописной). Переносы слов в заголовках не допускаются. Точку в конце заголовка не ставят.

шрифт TimesNewRoman, размер 14, все символы строчные кроме первой прописной;

выравнивание по левому краю с абзацным отступом;

межстрочный интервал одинарный;

интервал перед — **6**пт;

интервал после — **0** пт.

4.1.6 Оформление формул

В формулах в качестве символов следует применять обозначения, установленные соответствующими государственными стандартами. Пояснения символов и числовых коэффициентов, входящих в формулу, если они не пояснены ранее в тексте, должны быть приведены непосредственно под формулой. Пояснения каждого символа следует давать с новой строки в той последовательности, в которой символы приведены в формуле. Первая строка пояснения должна начинаться со слова «где» без двоеточия после него.

Формулы, следующие одна за другой и не разделенные текстом, разделяют запятой.

Формулы должны нумероваться в пределах раздела арабскими цифрами, которые записывают на уровне формулы справа в круглых скобках. Номер формулы состоит из номера раздела и порядкового номера формулы, разделенных точкой, например **(4.1).**

Пример оформления формулы:

$$V = \frac{4}{3} \times \pi \times R^3, \qquad (4.1)$$

где V– объем шара,

R– радиус шара.

Формулы в приложениях нумеруются в пределах каждого приложения с добавлением обозначения приложения - (В.1).

Ссылки в тексте на номера формул дают в скобках, например,«... в формуле (4.1)».

4.1.7 Оформление приложений

Приложения могут быть обязательными и информационными. Информационные приложения могут быть рекомендуемого или справочного характера.

Каждое приложение следует начинать с нового листа с указанием сверху в середине страницы слова "ПРИЛОЖЕНИЕ", напечатанного прописными буквами и его обозначения, а под ним в скобках для обязательного приложения пишут слово «обязательное», а для информационного - «рекомендуемое» или «справочное».

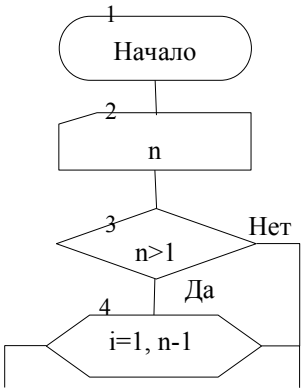
Приложение должно иметь содержательный заголовок, который размещается с новой строки по центру листа с прописной буквы.

Приложения обозначают заглавными буквами русского алфавита, начиная с А (за исключением букв Ё, З, И, О, Ч, Ь, Ы, Ъ), например: "ПРИЛОЖЕНИЕ А", "ПРИЛОЖЕНИЕ Б", "ПРИЛОЖЕНИЕ В". Допускается обозначать приложения буквами латинского алфавита, за исключением букв I и O. Если в документе одно приложение, оно обозначается "ПРИЛОЖЕНИЕ А".

Текст каждого приложения, при необходимости, может быть разделен на разделы и подразделы, которые нумеруются в пределах каждого приложения, при этом перед номером раздела (подраздела) ставится буква, соответствующая обозначению приложения (например: А.1.2 - второй подраздел первого раздела приложения А). Так же нумеруются в приложении иллюстрации, таблицы, формулы и уравнения.

Пример:

Пример фрагмента блок-схемы алгоритма сортировки массива



Иллюстрации и таблицы, помещаемые в приложении, нумеруют в пределах каждого приложения, например: «Рисунок А.2 - Форма «Склад»» (второй рисунок приложения А) или «Таблица К.1» (первая таблица приложения К).

Приложения должны иметь общую с остальной частью записки сквозную нумерацию страниц.

4.1.8 Оформление иллюстраций

Для пояснения текста могут быть приведены иллюстрации (фотографии, рисунки, чертежи, схемы, графики, таблицы), которые следует располагать как можно ближе к соответствующим частям текста.

Иллюстрации следует нумеровать в пределах раздела арабскими цифрами. Номер рисунка состоит из номера раздела и порядкового номера рисунка, разделенных точкой, например, «Рисунок 2.3». Если рисунок один, то он обозначается «Рисунок 1».

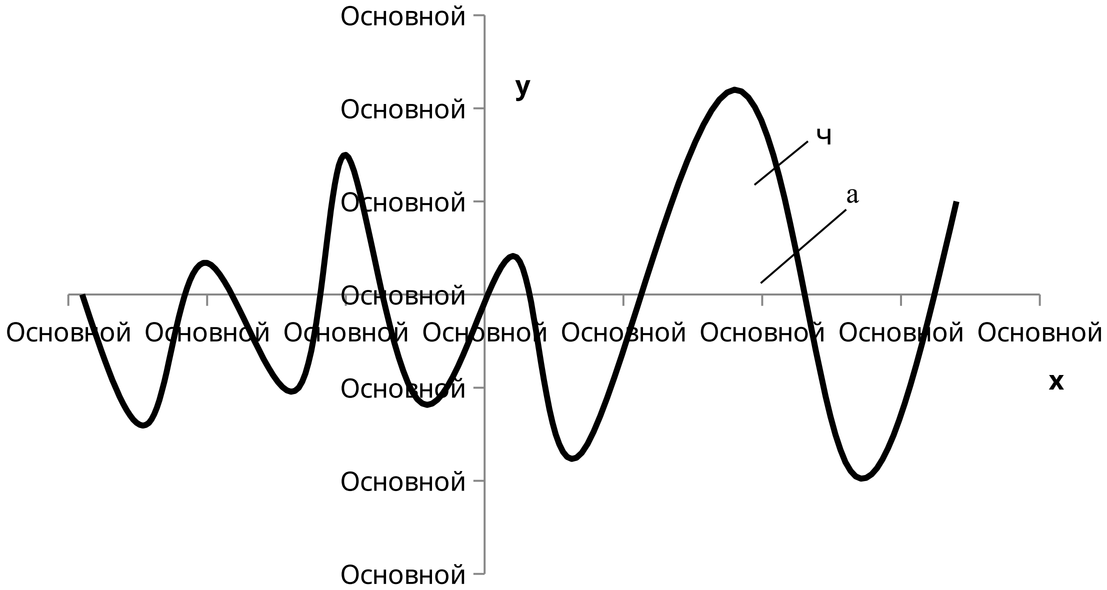
Иллюстрации каждого приложения обозначают отдельной нумерацией арабскими цифрами с добавлением перед цифрой обозначения приложения. Например, «Рисунок А.3». При ссылках на иллюстрации следует писать «... в соответствии с рисунком 2».

Иллюстрации должны иметь наименование и, при необходимости, пояснительные данные (подрисуночный текст).

Слово «Рисунок», номер и наименование помещают после рисунка и пояснительных данных (если имеются), например:

«Рисунок 1 - Виды оценки материальных ресурсов».

При необходимости между иллюстрацией и ее названием помещают поясняющие данные (подрисуночный текст), выполненный шрифтом размером на 1 пункт меньше, чем в основном тексте. Пример оформления иллюстраций приведен ниже.



а - аналитическое решение, ч - численное решение
Рисунок 4.4 - График функции φ (x,y)

4.1.9 Оформление таблиц

Таблицы применяют для лучшей наглядности и удобства сравнения показателей. Цифровой материал, как правило, должен оформляться в виде таблиц. Структурные элементы таблиц представлены на рисунке 4.5.

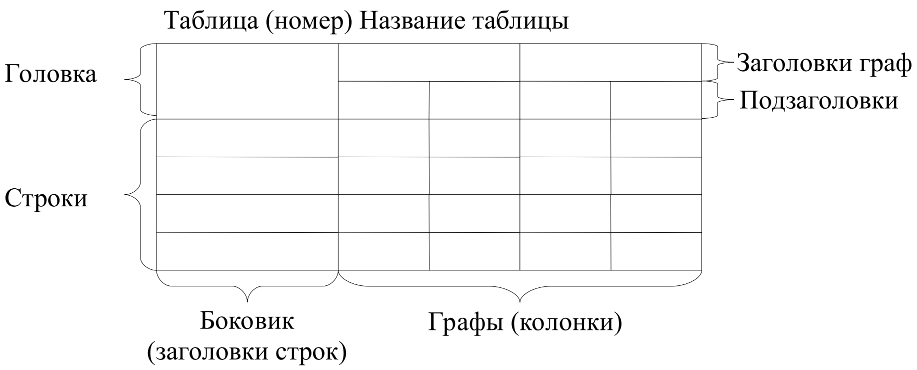


Рисунок 4.5 Структурные элементы таблицы

Каждая таблица должна иметь название, которое располагают над таблицей и выравнивают по левому краю таблицы. Название и слово «Таблица» начинается с прописной буквы. Например:

Таблица 2.1 - Результаты расчета параметров

Заголовки граф (колонок) и строк должны начинаться с прописных букв, подзаголовки - со строчных, если они составляют одно предложение с заголовком, и с прописных, если они самостоятельные. Делить заголовки таблицы по диагонали не допускается.

Высота строк должна быть не менее **8** мм.

Таблицы сверху, справа, слева и снизу ограничивают линиями.

Допускается применять размер шрифта в таблице меньший, чем в тексте.

Таблицу размещают после первого упоминания о ней в тексте таким образом, чтобы ее можно было читать без поворота работы или с поворотом по часовой стрелке.

Таблицу с большим количеством строк допускается переносить на другой лист. При переносе части таблицы на другой лист (страницу) слово «Таблица» и номер ее указывают только один раз слева над первой частью таблицы, над другими частями пишут слово «Продолжение»,

выравнивая по правой стороне таблицы. Если в работе несколько таблиц, то после слова «Продолжение» указывают номер таблицы, например: «Продолжение табл. 1.2». При переносе таблицы на другой лист наименование помещают только над ее первой частью, ниже заголовка должна следовать строка с номерами граф (колонок), которая располагается над частью таблицы на следующей странице.

Таблицу с большим количеством граф (колонок) допускается делить на части и помещать одну часть под другой в пределах одной страницы. Если строки или графы таблицы выходят за формат страницы, то в первом случае в каждой части таблицы повторяется ее головка, во втором случае — боковик.

Если цифровые или иные данные в какой-либо строке таблицы не приводят, то в ней ставят прочерк.

Таблицы, которые расположены на отдельных листах работы, включают в общую нумерацию страниц. Таблицы, размеры которых больше формата А4, учитывают как одну страницу и располагают в приложении.

Таблицы нумеруют последовательно (за исключением таблиц, приведенных в приложении) в пределах раздела. Номер таблицы должен состоять из номера раздела и порядкового номера таблицы, разделенных точкой, например: «Таблица 1.2» (вторая таблица первого раздела). Если в работе одна таблица, ее не нумеруют и слово «Таблица» не пишут.

На все таблицы, приведенные в пояснительной записке курсового проекта, должны быть ссылки в тексте, при этом слово «таблица» в тексте пишут полностью, если таблица не имеет номера, и сокращенно — если имеет номер, например: «...табл. 1.2» или «(табл. 1.2)».

4.1.10 Оформление ссылок

При написании пояснительной записки курсового проекта учащийся обязан давать ссылки на источники, материалы или отдельные результаты, которые приводятся в курсовой работе. Ссылки на использованные литературные источники должны включать в себя номер источника согласно списку использованных источников и помещаться в квадратные скобки. Например: [11, с. 54].

При использовании сведений, материалов из монографий, обзорных статей, других источников с большим количеством страниц в том месте пояснительной записки, где дается ссылка, необходимо указать номера страниц, иллюстраций, таблиц, формул, на которые дается ссылка в курсовой работе. Например: [9, с. 49, табл. 2] (здесь 9 - номер источника в списке, 49 - номер страницы, 2 - номер таблицы).

4.1.11 Оформление сносок

Если необходимо пояснить отдельные данные, приведенные в документе, то эти данные следует обозначать надстрочными знаками сноски.

Сноски располагают с абзацного отступа в конце страницы, на которой они обозначены, и отделяют от текста короткой тонкой горизонтальной линией с левой стороны, а к данным, расположенным в таблице - в конце ее над линией, обозначающей окончание.

Знак сноски ставят непосредственно после того слова, числа, символа, предложения, к которому дается пояснение, и перед текстом пояснения. Знак сноски выполняют арабскими цифрами со скобкой и помещают на уровне верхнего обреза шрифта (верхний индекс), например:

«... регуляризация$^{(2)}$...»

Нумерация сносок отдельная для каждой страницы.

Допускается вместо цифр выполнять сноски звездочками. Применять более четырех звездочек не рекомендуется.

4.1.12 Оформление примечаний

Примечания приводят в документах, если необходимы пояснения или справочные данные к содержанию текста, таблиц или графического материала.

Примечания помещают непосредственно после текстового, графического материала или в таблице, к которым относятся эти примечания, и пишут с прописной буквы с абзаца. Если примечание одно, то после слова «Примечание» ставят двоеточие и примечание пишется тоже с прописной буквы. Одно примечание не нумеруют. Несколько примечаний нумеруют по порядку арабскими цифрами. Номер примечания и текст разделяют пробелом, после номера точка не ставится.

4.1.13 Оформление примеров

Примеры могут быть использованы в тех случаях, когда текст требует конкретного или практического пояснения, или, когда необходимо, более краткое изложение.

Примеры размещают, нумеруют и оформляют так же, как и примечания.

4.2 Оформление графического материала

4.2.1 Общие положения

Общие правила выполнения чертежей регламентируются стандартами третьей группы ЕСКД (ГОСТ 2.301-68...ГОСТ 3.317-69).

На схемах, как правило, используются стандартные условные графические обозначения (УГО). При выполнении схем на больших форматах все УГО пропорционально увеличиваются по сравнению с приведенными в стандартах размерами. Размещение УГО на схеме должно обеспечивать наиболее простой рисунок схемы, с наименьшим числом изломов и пересечений линий связи, при сохранении между параллельными линиями расстояния не менее 3 мм. Линии связи и УГО выполняются линиями одной и той же толщины.

При необходимости на схемах помещается текстовая информация: наименования, обозначения, технические характеристики и т.п. Текстовые данные могут располагаться рядом с УГО (справа или сверху) или внутри УГО, рядом с линиями, в разрыве или в конце линий, на свободном поле схемы. Таблицы, помещаемые на свободном поле схемы, должны иметь наименование, раскрывающее их содержание.

На каждом листе графических документов, в том числе на листах с наглядными пособиями, выполняется рамка и основная надпись, которую располагают в правом нижнем углу листа вдоль длинной его стороны. Примеры выполнения основных надписей приведены в приложении Р. Форма, содержание, расположение и размеры граф должны соответствовать ГОСТ 2.104-68.

4.2.2 Оформление схем

Схемы всех типов как самостоятельные документы оформляются в соответствии со стандартами ЕСКД 2.701-2.711.

Схемы выполняют без соблюдения масштаба, действительное пространственное расположение составных частей объекта не учитывают или учитывают приближенно.

Графические обозначения элементов (устройств, функциональных групп) и соединяющие их линии связи следует располагать на схеме таким образом, чтобы обеспечивать наилучшее представление о структуре изделия и взаимодействии составных частей.

При выполнении схем применяют следующие графические обозначения, принятые в соответствующих стандартах или методиках. Допускается применять упрощенные внешние очертания, в том числе аксонометрические. При использовании нестандартных условных графических обозначений и упрощенных внешних очертаний на схеме приводят соответствующие пояснения.

Размеры условных графических изображений, а также толщины их линий должны быть одинаковыми на всех схемах для данного объекта. Допускается пропорционально изменять все размеры графических изображений.

Линии связи должны состоять из горизонтальных и вертикальных отрезков и иметь наименьшее количество изломов и взаимных пересечений.

Структурной называется схема, определяющая основные функциональные части объекта, их назначение и взаимосвязи. Структурные схемы предназначаются для общего ознакомления с разрабатываемым объектом.

Функциональные части на схеме изображают в виде прямоугольников, ромбов, овалов и других условных обозначений.

Графическое построение схемы должно давать наиболее наглядное представление о последовательности взаимодействия функциональных частей.

На линиях взаимосвязи рекомендуется стрелками обозначать направление хода процессов.

На схеме должны быть указаны наименования каждой функциональной части объекта.

При изображении функциональных частей в виде геометрических фигур наименования, типы и обозначения рекомендуется вписывать внутрь фигуры.

Допускается помещать на схеме поясняющие надписи, диаграммы или таблицы, определяющие последовательность процессов во времени.

Функциональной называется схема, разъясняющая определенные процессы, протекающие в отдельных функциональных частях системы. Функциональные схемы предназначаются для изучения принципов работы системы.

На функциональной схеме изображают функциональные части системы (элементы и функциональные группы), участвующие в процессе,

иллюстрируемом схемой, и связи между этими частями.

Функциональные части и связи между ними на схеме изображают в виде условных обозначений в соответствии с используемыми стандартами или методиками.

4.2.3 Блок-схемы алгоритмов

Блок-схемы алгоритмов являются основными чертежами программного дипломного проекта. Блок-схемы могут быть различных видов и различной степени детализации. Они могут отображать алгоритмы работы подпрограмм, порядок взаимодействия подпрограмм, последовательность обработки данных и т.д.

Блок-схемы алгоритмов должны выполняться согласно требованиям стандартов ЕСПД 19.003-80, 24.302-80 и др. (таблица 2.1).

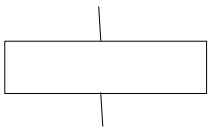
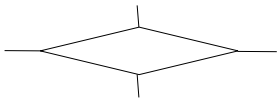
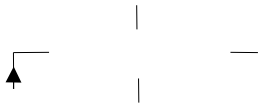
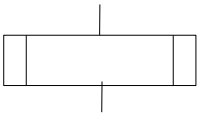
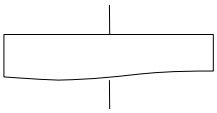
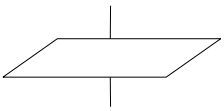
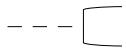
Основные моменты, на которые следует обратить внимание при черчении блок-схемы алгоритма:

1) Все вершины алгоритмов должны изображаться в виде специальных символов. Существует два основных стиля изображения символов, показанные на рисунке 4.6.

Рисунок 4.6 – Стили изображения символов блок-схемы алгоритма

Изображение символа зависит от функции, которую он отображает. Все символы в пределах чертежа должны вписываться в одинаковые виртуальные квадраты с соотношением высота: ширина, равным 2:3, и размерами 30x43 либо 40x60.

Таблица 2.1 – Условные графические изображения наиболее часто используемых блоков и элементов связей между ними

Название	**Элемент**	**Комментарий**
Процесс		Вычислительное действие или последовательность вычислительных действий
Решение		Проверка условия
Модификация		Заголовок цикла
Предопределенный процесс		Обращение к процедуре
Документ		Вывод данных, печать данных
Перфокарта		Ввод данных
Ввод/Вывод		Ввод/Вывод данных
Соединитель		Разрыв линии потока
Начало, Конец		Начало, конец, вход и выход во вспомогательных алгоритмах
Комментарий		Используется для размещения надписей

Входящие, по отношению к символам, линии обычно подводятся только сверху, а исходящие- снизу или со сторон. Например, для приведенного выше символа условной вершины алгоритма один вход расположен сверху, а два выхода с надписями - слева, справа либо снизу в любой комбинации.

Символ должен содержать поясняющий текст в соответствии с функцией (например условие X-Y) и координату на схеме (например А9). Поясняющий текст должен быть, по возможности, кратким и четким. Координата указывается в специальном разрыве контура символа, который делается в левом верхнем углу.

2) Символы-вершины алгоритмов объединяются с помощью символов-линий логической связи, показывающих информационные и другие потоки.

Линии логической связи могут иметь различную форму из параллельных линиям ограничительной рамки сегментов, разветвляться, объединяться и пересекаться произвольным образом. Разветвление указывается с помощью символа-точки. Примыкание одной линии связи к другой, а также направление логической связи укалываются с помощью символа-стрелки на конце последнего сегмента линии связи. Направления сверху вниз и справа налево считаются направлениями переходов по умолчанию. Поэтому если все сегменты линии связи направлены таким образом, то стрелка на конце линии *связи* не ставится. При сложной форме информационных потоков на блок-схеме стрелки расставляются таким образом, чтобы из любой наугад взятой точки

любой линии связи направлять движение в нужную сторону и чтобы общее число стрелок было минимальным.

3) Блок-схема алгоритма должна чертиться по координатной сетке. Высота ряда соответствует высоте символов блок-схемы, а ширина колонки - их ширине. Таким образом, допустимые варианты: 30 и 45 мм либо 40 и 60 мм.

Все символы вписываются в зоны координатной сетки. Координаты зон указываются для каждого из символов.

Рекомендуется начинать блок-схему в левой верхней части формата и продолжать ее вниз, а затем вправо.

Линии связи рисуются по виртуальной пятимиллиметровой сетке, привязанной к основной.

4) Следует иметь в виду, что существуют несколько исключений при изображении символов. Такие символы, как символы вершин начала и конца алгоритма, а также символы разрывов линий логической связи имеют половинную высоту и помещаются в верхние части соответствующих зон координатной сетки.

5) Блок-схема алгоритма может содержать комментарии, которые наносятся особым образом.

Комментарий изображается в виде особого символа - вертикальной (по возможности) квадратной скобки.

Символ комментария не привязывается к координатной сетке и, следовательно, не содержит координату. Символ комментария соединяется с комментируемым символом штриховой линией, не имеющей направления. Рекомендуется линию подводить к центру боковой стороны комментируемого символа и центру внешней (обязательно) стороны квадратной скобки. Текст комментария должен наноситься с внутренней стороны квадратной скобки и выравниваться по ней. Размер квадратной скобки должен соответствовать размеру текста.

Комментарий может относиться как к одному символу, так и к нескольким, возможно, выделенным в особый блок штрихпунктирной линией. У одного символа может быть несколько комментариев.

Рекомендуется помещать комментарии вблизи (слева или снизу) от тех символов, к которым они относятся.

6) При очень большом количестве пересечений линий логической связи, что встречается достаточно редко, а также при расположении одного чертежа на нескольких листах допускается делать разрывы линий связи, как показано на рисунке 4.7

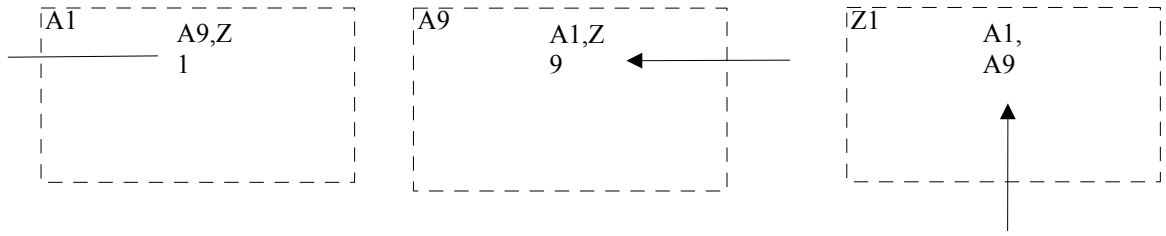


Рисунок 4.7 – Обозначение разрыва линии логической связи

Внутри особого символа разрыва перечисляются все координаты зон, где данная линия связи продолжается. Дополнительно могут указываться номера листов. Особенностью является то, что в одной зоне может присутствовать только один символ разрыва. Сокращения на блок-схемах алгоритмов встречаются редко.

Весь текст, содержащийся на блок-схеме алгоритма, должен быть нанесен шрифтом одного размера.

7) Рекомендуется использовать шрифт высотой 3,5 мм для символов с размерами 30х45 мм и 5 мм - для символов с размерами 40 х 60 мм. «Подгонка» текста к размерам символов делается за счет изменения плотности, а не высоты шрифта.

4.2.4 Оформление плакатов

При оформлении плакатов на оборотной стороне листа в правом нижнем углу выполняется основная надпись по форме для текстовых документов.

Форматная рамка не вычерчивается ни на оборотной, ни на лицевой сторонах листа Допускается использование стандартного штампа.

Рекомендуется использовать не более 4-х цветов и не более 6-ти толщин и типов линий.

Все элементы плаката (графические и текстовые) должны хорошо читаться с 4-5 метров.

Рекомендуемые размеры шрифтов:

- для заголовка плаката - высота букв не менее 5 см;
- для текста плаката - высота букв не менее 1,5 см.

Плакат обязательно должен иметь заголовок, отражающий его содержание. Заголовок рекомендуется начинать с существительного, например: Структура программного обеспечения, Характеристика используемых методов и алгоритмов обработки. Допускается начинать заголовок с определяющего прилагательного, например: Сравнительная характеристика..., Основные результаты...

Если материал плаката состоит из нескольких невзаимосвязанных частей, то каждая из них может иметь собственный заголовок или подрисуночную надпись, при этом обязателен общий объединяющий заголовок для плаката в целом.

4.2.5 Диаграммы

Для описания проекта информационной системы используются графические средства анализа и проектирования, обеспечивающие создание и редактирование иерархически связанных диаграмм (ОГО, ERD и др.), образующих модели информационных систем. Перечень наиболее распространенных диаграмм приведен ниже в таблице 2**.2.**

Диаграммы, как правило, являются основным способом отображения:

- взаимосвязей объектов (сущностей) в технологии реляционных баз данных;
- иерархии объектов (сущностей) при использовании объектно-ориентированного подхода при программировании.

Как правило, при построении диаграмм придерживаются принятых методик в системах проектирования. Как, например, в связи с отсутствием отечественных ГОСТов на изображение диаграмм классов рекомендуется использовать унифицированный язык моделирования UniversalModelingLanguage (UML), поддерживаемый многими системами программирования, например RationalRose.

Важным моментом является то, что диаграммы классов показывают отношения между классами, но не показывают порядок вызовов и взаимовлияние функций.

Таблица 4.2 – Типы диаграмм

Тип диаграммы	Обозначение
Сущность-связь	ERD
Потоков данных	DFD
Структур данных	DSD
Архитектуры системы	SAD
Типов данных	DTD
Потоков управления	CSD
Структуры меню	MSD
Последовательности блоков	BSD
Последовательности форм	FSD
Содержимого форм	FCD
Переходов состояний	STD
Структурных схем	SCD

Ключевые моменты, на которые следует обратить внимание:

1) На диаграмме класс (class) изображается в виде прямоугольника со сплошной границей, разделенного горизонтальными линиями на 3 основные секции.

Верхняя секция содержит имя класса и другие общие свойства, например стереотип. Если класс является абстрактным, то его имя приводится курсивом. Средняя секция содержит список атрибутов, то есть данных, инкапсулированных в класс. Нижняя секция содержит список операций, то есть функций-методов класса. Элементы этих списков можно группировать по некоторым признакам, причем в таких случаях перед группой ставится заключенная в кавычки строка, определяющая общее свойство.

2) Атрибут (attribute) изображается в виде текстовой строки, отражающей различные его свойства:

<видимость><имя>:<тип>=<начальное значение>{<свойства>}

3) Операция (operation) также изображается в виде текстовой строки:

< видимость><имя> (<список_параметров>): <тип возвращаемого значгния>{<своиства>)

4) «Видимость» имеет C++ семантику:

- открытый атрибут или открытая операция (public) - обычно обозначается символом +;
- защищенный атрибут или защищенная операция (protected) -обычно обозначается символом #;
- закрытый атрибут или закрытая операция (private) - обычно обозначается символом -.

5) Отношения между классами показываются с помощью различных видов линий и стрелок:

- отношение ассоциации (association), то есть связи вообще: один класс каким-либо образом связан с другим классом – обозначается обычнойлинией без стрелки, возле которой могут быть дополнительные имя ассоциации, тип ассоциации, количество участвующих в ассоциацииобъектов от каждой из сторон и др.);
- отношение группировки (aggregation), то есть владения: один класс входит в другой класс - обозначается обычной линией и не закрашенной стрелкой-ромбом со стороны класса-владельца;
- отношение слияния (composition), то есть «сильного» владения- один класс входит только в определенный другой класс . обозначается обычной линией и закрашенной стрелкой-ромбом со стороны класса-владельца;
- отношение детализации (datailization), то есть использования: один класс «реализует» другой класс - обозначается пунктирной линией и незакрашенной стрелкой-треугольником со стороны класса-пользователя;
- отношение зависимости (dependency), то есть влияния: <<модификация>> одного класса влияет на другой класс - обозначается пунктирной линией и стрелкой-углом со стороны класса-потребителя;
- отношение обобщения (generalization), то есть наследования: один класс (производный класс) является «частным случаем» другого класса (базовый класс) - обозначается обычной линией и незакрашенной стрелкой-треугольником со стороны класса-родителя.

Более подробные сведения излагаются в спецификации UML.

4.3 Оформление исходных текстов программ

4.3.1 Общие требования к оформлению листингов

Пример оформления листинга программы приведен в приложении М. Если в тексте приводится несколько фрагментов программного кода, то им присваивают заголовки. Заголовок листинга выравнивают по левому краю, например:

Листинг 1 - Модуль ListOperations.cpp

Для оформления листингов используют шрифт CourierNew.

4.3.2 Форматирование

Выравнивание и отступы

В качестве отступов используется два пробела, табуляция по Tab запрещена.

Зарезервированные слова *unit, uses, type, interface, implementation, initialization u finalization* всегда должны примыкать к левой границе. Также должны быть отформатированы финальный *end u end,* завершающий исходный модуль.

В файле проекта выравнивание по левой границе применяется к словам *program,* главным *begin u end*

Использование пробелов и пустых строк

Пустые строки могут повысить читабельность путем группирования секций кода, которые логически связаны между собой.

Пустые строки должны использоваться в следующих местах:

- после блока копирайта;
- после декларации пакета;
- после секции импорта;
- между объявлениями классов;
- между реализациями методов.

Пробелы должны использоваться:

- после запятой при разделении параметров при вызове функции;
- между бинарными операторами.

Запрещено использовать пробелы:

- перед двоеточием при объявлении переменных;
- до или после оператора .(точка);
- между именем метода и открывающей скобкой;
- между унарным оператором и его операндом;
- между операторами приведения и приводимым выражением;
- после или перед открывающейся скобкой (круглой или квадратной);
- перед точкой с запятой.

4.3.3 Копирайт

Копирайт помещается в начале модуля до объявления имени модуля.

Примерный вид блока копирайта:

```
{=========================================================}

{ Модуль XXX }

{ гр. П414}

{ Разработчик: Иванов И.В }

{ Модифицирован: 7 июня 2017 }

{----------------------------------------------------------------------------------------------------)

{ Краткое назначение модуля. Например }

{ Модуль, обеспечивающий чтение/запись базы данных} {из/в файл }

{**************************************************************************************}
```

Пример оформления блок-схемы алгоритма

ПРИЛОЖЕНИЕ Б

Примерные темы курсовых работ

1Разработка функции хэширования

2 Разработка локального чата

3Разработка алгоритма шифрования RSA

4Разработка инженерного калькулятора

5 Разработка игры Точки

6 Разработка игры Жизнь

7 Разработка интерпритатора языка программирования

8 Разработка сетевого калькулятора

9 Разработка алгоритма шифрования DES

10 Разработка распределенного приложения на базе блокчейн технологии «Учет денежных переводов физических лиц»

11 Разработка системы на базе блокчейн технологии «Безопасное дорожное движение»

12 Разработка системы на базе блокчейн технологии «Система голосования»