# Введение

Ведение народного хозяйства страны невозможно без детального изучения ее территории. Современная техника геодезических и съемочных работ обеспечивает получение точной топографической карты, характеризующей географический ландшафт по его основным элементам. Эта карта, кроме непосредственного обслуживания народнохозяйственных нужд, дает еще основу для изучения страны в геологическом, гидрогеологическом, почвенном, ботаническом и других отношениях. На основе этой карты составляются различные общегеографические и специальные карты.

В науках о Земле информационные технологии породили геоинформатику и географические информационные системы (ГИС). Любая ГИС представляет из себя систему для сбора, хранения, анализа и представления картографической информации. Геоинформационные системы могут включать природную, биологическую, культурную, демографическую или экономическую информацию.

Для использования в ГИС данные должны быть преобразованы в подходящий цифровой формат. Процесс преобразования данных с бумажных карт в компьютерные файлы называется оцифровкой. В современных ГИС этот процесс может быть автоматизирован применением сканерной технологии, что особенно важно при выполнении крупных проектов. Если объем работ небольшой, можно вводить данные с помощью дигитайзера. Многие данные уже переведены в форматы, напрямую воспринимаемые ГИС-пакетами.

ГИС хранит информацию о реальном мире в виде набора тематических слоев, которые объединены на основе географического положения.

ГИС может работать с двумя существенно различающимися типами данных – векторными и растровыми. В векторной модели информация о точках, линиях и полигонах кодируется и хранится в виде набора координат X и Y.

Растровая модель оптимальна для работы с непрерывными свойствами. Растровое изображение представляет собой набор значений для отдельных элементарных составляющих (ячеек). Оно подобно отсканированной карте или картинке. Вся изучаемая территория разбивается на элементы регулярной сетки или ячейки. Каждая ячейка содержит только одно значение. Она является пространственно заполненной, поскольку любое местоположение на изучаемой территории соответствует ячейке растра, иными словами, растровая модель оперирует элементарными местоположениями. В большинстве растровых моделей данных наименьшей единицей является квадрат или прямоугольник. Такие единицы известны как сетка, матрица или пиксел. Множество ячеек образует решетку, растр, матрицу.

Бурное развитие геоинформатики порождает немало приложений, в которых используется та или иная цифровая модель рельефа (ЦМР) земной поверхности. Подавляющее число таких приложений, например, прогноз наводнений, обеспечение безопасности полетов, объемное представление карты на экране компьютера и т.п., требуют по возможности наиболее точных моделей. Зачастую требования по точности ЦМР можно удовлетворить, создавая модели на основе цифровых топографических карт соответствующего масштаба, содержащих информацию о рельефе в виде изолиний, отметок высот, отметок урезов воды и т.п. Процесс создания модели рельефа (в триангуляционном или матричном виде) по цифровым данным такого типа в настоящее время хорошо изучен и реализован во многих ГИС-пакетах.

Цифровая модель рельефа, ЦМР – это средство цифрового представления 3-мерных пространственных объектов (поверхностей, рельефов) в виде трехмерных данных как совокупности высотных отметок или отметок глубин и иных значений аппликат (координаты Z) в узлах регулярной сети с образованием матрицы высот, нерегулярной треугольной сети или как совокупность записей горизонталей (изогипс, изобат) или иных изолиний.

Целью дипломного проекта является разработка программного модуля обработки и отображения картографической информации на цифровой карте местности.

В процессе дипломного проектирования решаются следующие задачи:

* обзор технологий и систем ГИС;
* системное и функциональное проектирование программного модуля;
* разработка программного модуля с использованием сред программирования Visual C++ 6.0, Qt 3.3.3;
* проведение тестирования программного модуля;
* разработка руководства пользователя программного модуля;
* технико-экономическое обоснование;
* разработка раздела по охране труда.

геоинформационный программирование модуль система

# Обзор геоинформационных систем

## 

## 1.1 Понятие о геоинформационных системах и технологиях

Создание карт и географический анализ не являются чем-то абсолютно новым. Технологии ГИС предоставляют новый, более соответствующий современности, более эффективный, удобный и быстрый подход к анализу проблем и решению задач, стоящих перед человечеством в целом, и конкретной организацией или группой людей, в частности. Они автоматизируют процедуру анализа и прогноза. До начала применения ГИС лишь немногие системы обладали искусством обобщения и полноценного анализа географической информации с целью обоснованного принятия оптимальных решений, основанных на современных подходах и средствах.

Основное отличие ГИС от просто информационной системы в том что, ГИС – информационная система, обеспечивающая сбор, хранение, обработку, доступ, отображение и распространение пространственно координированных (пространственных) и атрибутивных данных. ГИС содержит данные о пространственных объектах в форме их цифровых представлений (векторных, растровых), включает соответствующий задачам набор функциональных возможностей ГИС, в которых реализуются операции геоинформационных технологий, поддерживается программным, аппаратным, информационным, нормативно-правовым, кадровым и организационным обеспечением. ГИС различаются предметной областью информационного моделирования, к примеру, городские ГИС, или муниципальные ГИС, природоохранные ГИС, среди них особое наименование, как особо широко распространенные, получили земельные информационные системы. Проблемная ориентация ГИС определяется решаемыми в ней задачами (научными и прикладными), среди них инвентаризация ресурсов (в том числе кадастр), анализ, оценка, мониторинг, управление и планирование, поддержка принятия решений. Интегрированные ГИС совмещают функциональные возможности ГИС и систем цифровой обработки изображений (данных дистанционного зондирования) в единой интегрированной среде. Пространственно-временные ГИС оперируют пространственно-временными данными. Реализация геоинформационных проектов, создание ГИС в широком смысле слова, включает этапы: исследования, технико-экономическое обоснование, оценку соотношения «затраты / прибыль», системное проектирование ГИС, тестирование на небольшом территориальном фрагменте, создание опытного образца, внедрение ГИС в эксплуатацию и использование. Научные, технические, технологические и прикладные аспекты проектирования, создания и использования ГИС изучаются наукой геоинформатикой.

Хотя большинство ГИС преимущественно используются управления объектами на сравнительно больших территориях, однако, они могут быть применены и предприятиями, имеющих на своем балансе здания, сооружения, инженерные коммуникации. Для таких систем хорошо подходят корпоративные ГИС. Задача любой корпоративной системы – снизить издержки работы за счет наведения порядка в том, с чем до этого работали на бумаге, ускорить доступ к информации, автоматизировать проведение расчетов, которые на компьютере производятся, естественно, быстрее и с меньшей вероятностью возникновения ошибки, чем на бумаге. Для получения лучших результатов такие системы должны обладать интеллектуальными свойствами.

Работающая ГИС включает в себя пять ключевых составляющих: аппаратные средства, программное обеспечение, данные, исполнители и методы [1].

Аппаратные средства, которые используются в ГИС – это компьютер, на котором запущена ГИС и средства ввода-вывода пространственной информации (дигитайзеры, плоттеры и др.). В настоящее время ГИС работают на различных типах компьютерных платформ, от централизованных серверов до отдельных или связанных сетью настольных компьютеров.

Программное обеспечение ГИС содержит функции и инструменты, необходимые для хранения, анализа и визуализации географической (пространственной) информации. Ключевыми компонентами программных продуктов являются: инструменты для ввода и оперирования географической информацией; СУБД; инструменты поддержки пространственных запросов, анализа и визуализации (отображения); ГИП для легкого доступа к инструментам.

Данные – это вероятно наиболее важный компонент ГИС. Данные о пространственном положении объектов (географические данные) и связанные с ними табличные данные могут собираться и подготавливаться самим пользователем, либо приобретаться у поставщиков на коммерческой или другой основе. В процессе управления пространственными данными ГИС интегрирует пространственные данные с другими типами и источниками данных, а также может использовать СУБД, применяемые многими организациями для упорядочивания и поддержки имеющихся в их распоряжении данных.

Широкое применение технологии ГИС невозможно без людей (исполнителей.), которые работают с программными продуктами и разрабатывают планы их использования при решении реальных задач. Пользователями ГИС могут быть как технические специалисты, разрабатывающие и поддерживающие систему, так и обычные сотрудники (конечные пользователи), которым ГИС помогает решать текущие каждодневные дела и проблемы.

Успешность и эффективность (в том числе экономическая) применения ГИС во многом зависит от правильно составленного плана и правил работы, которые составляются в соответствии со спецификой задач и работы каждой организации.

В настоящее время российские и зарубежные производители предлагают широкий спектр геоинформационных систем. Можно привести наиболее популярные из них.

GIS MapInfo – содержит всю информацию (графическую, текстовую и др.) в таблицах. Пользователь управляет процессами создания картографической информации и тематических слоев через графический интерфейс, который выражается в виде меню и системы контекстных подсказок. MapInfo имеет встроенную реляционную СУБД, работающую по SQL-запросам с файлами в различных форматах.

BusinessMAP – пакет начального уровня для решения бизнес задач, таких как поддержка продаж и маркетинг, на основе включенных в него (в основном по США) или уже имеющихся в готовом виде данных.

Atlas GIS – удобный пакет для конечного пользователя со всеми стандартными функциями, имеющимися в простых настольных ГИС системах. Включает разнообразные готовые данные, в том числе по России. Основное назначение – бизнес приложения.

ArcView GIS – настольная ГИС с развитыми функциями пространственного анализа, многие из которых добавляются к базовому набору в виде дополнительных модулей. Работает на разных платформах.

ARC/INFO – профессиональная ГИС для решения задач структурирования и управления данными, ведения базы данных. Работает на разных платформах. ГИС для рабочей группы (уровня отдела или небольшой фирмы, в которых постоянно ведется разработка ГИС проектов). По мере расширения сферы деятельности и усложнения выполняемых задач целесообразно распределить их решение между несколькими продуктами. Такая ГИС обычно базируется на разделяемой базе пространственных данных, поддерживаемой и обновляемой с помощью разных продуктов и обеспечивающей работу «конечных» пользователей на персональных компьютерах в пределах данной рабочей группы или отдела. Конкретное предлагаемое решение зависит от типов используемых данных, выбранной программной среды, аппаратного обеспечения и ряда других факторов. ГИС уровня предприятия (корпоративная ГИС). Обеспечивает всю организацию пространственной информационной поддержкой. В такую ГИС можно встроить сотни специализированных приложений с дополнительными средствами пространственных запросов, расширенного анализа, отображения и представления данных. В качестве компонентов системы такого уровня могут использоваться серверы пространственных данных:

Spatial Database Engine (SDE) – открытая среда управления пространственными данными, ядро корпоративной распределенной ГИС, как правило, вместе с внешней реляционной СУБД;

ГИС широко используется в интернете. Глобальное картографическое решение. ESRI предоставляет два стандартных решения задачи публикации карт в сети Интернет. MapObjects Internet Map Server обеспечивает технологические средства для доступа к базе данных ГИС через Интернет, разработки собственных приложений для создания карт по имеющимся в вашей организации данным, поддержки пространственных запросов к Web серверу с любого места в сети. ArcView Internet Map Server предоставляет в ваше распоряжение готовые средства для распространения картографических материалов через Интернет без необходимости дополнительного программирования.

Как видно из вышеизложенного материала в настоящее время ГИС – средства в полной мере не решают интеллектуальных задач, таким образом как их решает человек. Основная проблема сводится к манипулированию и использованию знаний. Современные ГИС позволяют работать только с данными (пространственными и атрибутивными) но в полной мере не позволяют представить знания о предметной области, знания как управлять предприятием. Одним из путей решения данной проблемы является реализация системы «гибрида» существующих разработок реализация подсистемы осуществляющей работу с пространственными данными плюс уже более новыми и предметно ориентированными (представление знаний) средствами для представления и работы с атрибутивными данными и знаниями об исследуемых объектах – подсистемой поддержки и принятия решений.

Целью данного дипломного проекта является разработка программного модуля, который должен выполнять следующие функции (задачи):

* загрузка, отображение цифровой карты местности (ЦКМ), прокрутка и зуммирование изображения загруженной цифровой карты местности (с использованием функций существующей библиотеки картографического обеспечения);
* создание, редактирование и сохранение списка стандартных высот для объектов цифровой карты местности (по видам объектов, согласно классификатора условных знаков);
* расчет матрицы превышений на фоне цифровой карты местности (с учетом списка стандартных высот, соответствующих данному участку местности);
* создание и редактирование матрицы корректировок.

Задача расчета матрицы превышений может решаться как однократно для вполне определенных заданных значений высот рельефа местности, растительности, гидрографии, промышленных объектов, населенных пунктов т. п., так и многократно, а также по мере необходимости при изменении:

* данных о рельефе, гидрографии, растительности;
* положения и состояния промышленных объектов, автомобильных и железных дорог или населенных пунктов.

Исходными данными для программного модуля являются:

* несколько файлов цифровой карты местности с расширением UTP;
* файл – классификатор условных знаков объектов местности с расширением KLS;
* файл стандартных высот объектов местности (по видам объектов, согласно классификатора условных знаков);
* файл матрицы высот с расширением BMH;
* файл матрицы превышения с расширением BMH2.

Форматы файлов UTP, KLS, BMH, BMH2 являются внутренними для существующей библиотеки картографического обеспечения. Библиотека содержит все необходимые функции для работы с файлами указанных форматов.

Файлы цифровой карты местности содержат пространственные, количественные, качественные и описательные характеристики объектов. Один из предоставляемых файлов является базовым, т.е. покрывает весь интересующий регион и содержит минимальный набор объектов. Остальные файлы ЦКМ соответствуют листам карт масштаба 1:200000 или 1:500000.

Объекты карты распределены по слоям (гидрография, границы и т.д.). Отображение карты производится последовательно слой за слоем в соответствии с классификатором.

Файл классификатора объектов карты содержит данные об их представлении на экране при отображении ЦКМ в зависимости от масштаба отображения.

Файл матрицы высот содержит данные о возвышении земной (водной) поверхности относительно уровня Балтийского моря в каждой точке местности. Данные содержатся в матричном виде, в файле также хранится информация о размере ячейки (шага матрицы), регионе охвата.

Файл превышения высот содержит данные о возвышении верхней поверхности сооружений, растительного покрова относительно земной поверхности в каждой точке местности. Формат файла матрицы превышений такой же, как у файла матрицы высот.

Файл стандартных высот объектов местности представляет из себя список возможных видов объектов (согласно используемого классификатора условных знаков) и соответствующих им значений стандартных высот, хранимый в виде обычного текстового файла.

Для улучшения зрительного восприятия слоев и ускорения процесса их отображения можно применять функции упрощенной отрисовки объектов из существующей библиотеки картографического обеспечения.

Степень детализации отображаемой карты может изменяться установкой минимального размера отображаемого объекта (предусмотреть соответствующие элементы интерфейса). Размер задается в единицах измерения экранной системы координат, т.е. в пикселях.

Для реализации функций загрузки, отображения, прокрутки и зуммирования ЦКМ предусмотреть соответствующие элементы интерфейса (например, меню главного окна или кнопки).

Разрабатываемый программный модуль должен функционировать в среде операционной системы Microsoft Windows 2000/XP, желательно также предусмотреть при разработке интерфейса возможность функционирования программы в операционной системе RedHat Linux 7.3/8. Поэтому предлагается интерфейс приложения разрабатывать с использованием библиотеки QT (версии 3.1.2/3.3.3).

# Системное проектирование

## 

## 2.1 Разработка структуры программного модуля

Исходя из требований постановки задачи, разрабатываемый программный модуль обработки и отображения картографической информации можно структурно разбить на несколько блоков:

* блок, реализующий функции загрузки и отображения цифровой карты местности с использованием библиотеки картографии, разработанной специалистами УП НИИСА. Входными данными используемых функций библиотеки картографии являются:
* один или несколько файлов цифровой карты местности с расширением UTP;
* файл – классификатор условных знаков объектов местности с расширением KLS;
* файл матрицы высот с расширением BMH.
* блок, отвечающий за создание, редактирование и сохранение списка стандартных высот для объектов цифровой карты местности. Внутри блока реализуются также визуальные средства (окно) для редактирования списка высот. Результатом работы данного блока является текстовый файл с расширением ТХТ.
* блок, необходимый для расчета и сохранения матрицы превышений на фоне цифровой карты местности. Сохранение результатов расчета происходит в двоичный файл с расширением BMH2.
* блок для создания и редактирования матрицы корректировки. Результатом работы данного блока является файл с расширением BMH3.

Обобщенная структурная схема программного модуля представлена на черт. РТДП 5.000.002.

Разработка пользовательского интерфейса программы будет осуществляться с помощью средств библиотеки Qt.

GUI инструментарий, именуемый Qt, предоставляет норвежская компания TrollTech. GUI означает «Графический Пользовательский Интерфейс» («Graphical User Interface»), поэтому базирующиеся на Qt приложения представляются кнопками, окнами и т.п. Такие приложения осуществляют взаимодействие с пользователем, представляя в визуальной форме выполняемые функции. Этот инструментарий необходим для разработки графических приложений, которые используют X-Window интерфейс в Unix системах, потому что X не содержит предопределенного пользовательского интерфейса. И хотя другие наборы инструментов также позволяют создавать пользовательский интерфейс, Qt предоставляет некоторые технические возможности, которые делают разработку приложений очень простой. Кроме того, Qt инструментарий также доступен для Microsoft Windows систем, что позволяет разработчикам выпускать приложения для обеих платформ.

В настоящее время Qt – это библиотека классов C++ и набор инструментального программного обеспечения, предназначенных для построения многоплатформенных приложений с графическим интерфейсом и исповедующих принцип «написав однажды – компилируй в любом месте». Qt представляет собой единую платформу для приложений, которые могут работать под управлением Windows 95/98/Me/2000/XP, Mac OS X, Linux, Solaris, HP-UX и других версий Unix.

Qt предоставляет:

* широкий набор классов и методов, пригодных для использования даже для программ без графического интерфейса;
* хорошее решение для реализации взаимодействия с пользователем посредством виртуальных методов и механизма сигнал / слот;
* набор предопределенных GUI-элементов, называемых «widgets», которые могут быть легко использованы для создания видимых элементов;
* дополнительные полностью предопределенные диалоговые окна, которые часто используются в приложениях, такие как окно прогресса или окно выбора файлов.

В качестве языка программирования выбран C++ как язык, наилучшим образом сочетающий в себе поддержку объектно-ориентированной технологии программирования и возможности написания кода на достаточно низком уровне, что позволяет достичь оптимального быстродействия разрабатываемого программного обеспечения. Одним из современных инструментальных средств для разработки Windows-приложений является пакет Microsoft Visual C++ 6.0. Visual C++ предоставляет удобную среду визуального объектно-ориентированного программирования, что и обусловило выбор именно этого пакета в качестве среды разработки.

## 2.2 Логическая структура библиотеки картографии

Библиотека картографии предназначена для работы с цифровой картой местности (ЦКМ) и представляет собой набор классов и функций, обеспечивающих визуализацию и работу с картой, а также обеспечение решения расчетных задач на фоне карты.

Программное обеспечение состоит из следующих элементов:

* набор классов и функций, составляющих ядро библиотеки;
* класс MKartaInterface, предоставляющий функции управления картой.

Для хранения карты на диске используются файлы в формате UTP. Формат UTP предполагает наличие utp-файла, а также kls-файл, содержащий классификатор. Также возможно использование bmh-файла с матрицей высот.

Логически карта состоит из паспорта, ссылки на классификатор (в виде термина) и одного или нескольких слоёв, каждый из которых, в свою очередь, состоит из объектов.

Термин классификатора указывает классификатор, который содержит знаковую систему, необходимую для формирования корректного изображения карты. Карты не являются самостоятельными, независимыми от классификаторов единицами. Карта может быть загружена даже в случае отсутствия файла классификатора, но при этом вид карты на экране будет схематичным.

Слой – самая крупная структурная единица в структуре карты, представляющая собой хранилище объектов. Структурно состоит из паспорта слоя и списка объектов.

В картографии понятие «объект» обозначает наиболее элементарную часть в структуре карты, соответствующую единичному географическому явлению на местности.

Как информационная структура, объект представляет собой хранилище сведений об изображаемом явлении, включающее:

* указание на способ изображения данного явления, т.е. его условный знак;
* данные о расположении явления (пространственные данные, контур);
* дополнительные сведения о нем (атрибуты).

# 

# Функциональное проектирование

## 

## 3.1 Проектирование структуры классов

Исходя из выделенных на этапе структурного проектирования блоков разрабатываемого программного модуля, для реализации заданных функции были разработаны описанные ниже классы.

Класс MainWindow – описывает главное окно программного модуля.

Класс MapView – реализует отображения карты в главном окне программного модуля.

Класс Tasks – обрабатывает нажатие функциональных клавиш в главном окне программного модуля.

Класс Heights – реализует работу со списком стандартных высот.

Класс EditDialog – описывает работу диалога редактирования стандартных высот.

Класс Matrix – реализует работу с матрицей превышений и матрицей корректировок.

Класс ComputeDialog – описывает работу диалога расчета матрицы превышений для нескольких листов карт местности.

Класс ComputeThread – реализует процедуру расчета матрицы превышений для нескольких листов карт местности.

Взаимодействие всех классов осуществляется с помощью следующих глобальных переменных объявленных в main.cpp как указатели на объекты соответствующих классов:

* MKartaInterface \*karta – объект, реализующий функции взаимодействия с картой;
* Matrix \*matrix – текущая обрабатываемая матрица превышений;
* Matrix \*correctionMatrix – текущая обрабатываемая матрица корректировок;
* MainWindow \*window – главное окно приложения;
* Tasks \*tasks – объект для обработки нажатия клавиш в главном окне.

### Классы, реализующие диалоги программного модуля

Класс MainWindow – с помощью этого класса происходит описание главного окна программного модуля. Основные функции данного класса:

* MainWindow() – конструктор, который вызывается при создании окна;
* bool eventFilter (QObject \*o, QEvent \*e) – функция обработки движения мыши;
* void onExit() – функция выполняет последовательность операций при закрытии главного окна программы.

Класс EditDialog – с помощью этого класса происходит описание диалога редактирования стандартных высот. Основные функции этого класса:

* EditDialog() – конструктор, который вызывается при создании диалога;
* void addButtonClick() – функция обработки добавления элементов в список;
* void remButtonClick() – функция обработки удаления элементов из списка;
* void listViewClick (QListViewItem \*item) – функция обработки выбора раздела классификатора;
* void updateTables (QString zs, QString razd) – функция обновления списка высот для знаков выбранного раздела классификатора;
* void valueChanged (int row, int col) – функция обработки изменения значения высоты в списке.

Класс ComputeDialog – с помощью этого класса происходит описание диалога расчета матрицы превышений для нескольких листов карт местности. Основные функции данного класса:

* ComputeDialog (QWidget\* parent = 0, const char\* name = «ComputeDialog», bool modal = FALSE, WFlags fl = 0) – конструктор, который вызывается при создании диалога;
* void start() – функция обработки запуска процедуры расчета;
* void stop() – функция обработки остановки процедуры расчета;

При реализации пользовательского интерфейса использованы стандартные элементы управления Windows, каждому из которых в Qt соответствует собственный класс:

* панель управления – используется для ускоренного доступа к часто используемым функциям – класс QToolBar;
* главное меню – используется для организации меню – класс QMenuBar;
* панель состояния – используется для вывода сервисной информации – класс QStatusBar;
* список – представление данных в виде списка – класс QListView;
* надпись – используется для подписывания элементов интерфейса – класс QLabel;
* поле ввода – используется для ввода данных пользователем – класс QLineEdit;
* кнопка – используется для выполнения пользователем определенных действий – класс QPushButton;
* индикатор прогресса – класс QProgressBar.

Кроме того, использованы некоторые специальные классы Qt:

* QMainWindow – для создания главного окна приложения (класс MainWindow);
* QTable – для организации таблиц с данными;
* QDialog – для организации окон модальных диалогов (классы EditDialog, ComputerDialog).

С помощью программы Qt Designer, входящей в комплект поставки системы разработки Qt, был разработан вид главного окна программного модуля и диалогового окна для редактирования стандартных высот.

Программа Qt Designer предназначена для разработки и реализации графического пользовательского интерфейса приложения. С ее помощью также можно сгенерировать программный код, обеспечивающий реализацию функций разработанного интерфейса. В отличие от Microsoft Visual Studio, помещающего описание ресурсов диалогового окна в текстовый файл с расширением.rc, среда разработки Qt создает файл с расширением.ui и помещает в него описание диалогового окна в формате, совместимом с XML. Этот файл включается в makefile данного проекта, создающий на его основе соответствующие файлы заготовок и реализации класса диалогового окна.

Вид главного окна приложения с описанием элементов представлен на рисунке 3.1.



Рисунок 3.1. Главное окно программы

Вид диалогового окна для редактирования списка стандартных высот с описанием элементов представлен на рисунке 3.2.



Рисунок 3.2. Диалоговое окно для редактирования списка стандартных высот

### Классы, реализующие функции программного модуля

Класс ComputeThread – с помощью этого класса происходит расчет матрицы превышений для нескольких листов карт местности. Основные функции этого класса:

* ComputeThread (QObject \*obj) – конструктор, который вызывается при создании объекта;
* void start() – функция запуска процедуры расчета;
* void stop() – функция остановки процедуры расчета;
* void run() – функция выполнения процедуры расчета.

Класс Heights – с помощью этого класса происходит работа со списком стандартных высот. Основные функции этого класса:

* void save (MString name) – функция сохранения в файл;
* bool load (MString name) – функция загрузки из файла;
* void clear() – функция обеспечивающая удаление всех объектов, очистку списка;
* iterator find (const HObj &o) – функция обеспечивающая поиск объекта в списке;
* void insert (const HObj &o) – функция обеспечивающая вставку объекта в список;
* void erase (const HObj &o) – функция обеспечивающая удаление объекта;

Структура HObj реализует объекты составляющие список стандартных высот. Для каждого объекта хранятся код, название и высота (превышение). Конструктор обеспечивает первоначальную инициализацию данных.

Класс MapView – с помощью этого класса реализуется отображение карты в главном окне программного модуля. Основные функции этого класса:

* mousePressEvent() – функция обработки нажатие мыши при построении матрицы корректировки;
* void zoom (int left, int top, int right, int bottom) – функция обработки процедуры зуммирования;
* void prevView() – функция обработки процедуры возврата к предыдущему виду;
* void defaultView() – функция обработки процедуры отображения всей карты;
* void updateHeights() – функция осуществляющая перерисовку (отображение) матриц превышений и корректировки на текущем отображаемом фрагменте карты;
* void drawMap() – функция обработки процедуры отрисовки карты

Класс Tasks – с помощью этого класса происходит обработка нажатия функциональных кнопок в главном окне программного модуля. Основные функции этого класса:

* void loadMap() – функция, обрабатывающая нажатие кнопки открыть карту;
* void openFolder() – функция, обрабатывающая нажатие кнопки открыть папку;
* void loadKls() – функция, обрабатывающая нажатие кнопки открыть классификатор;
* void loadHeights() – функция, обрабатывающая нажатие кнопки открыть список стандартных высот;
* void loadMatrix() – функция, обрабатывающая нажатие кнопки открыть матрицу превышений;
* void loadCorrectionMatrix() – функция, обрабатывающая нажатие кнопки открыть матрицу корректировок;
* void newHeights() – функция, обрабатывающая нажатие кнопки создать список стандартных высот;
* void newCorrectionMatrix() – функция, обрабатывающая нажатие кнопки создать матрицу корректировок;
* void editHeights() – функция, обрабатывающая нажатие кнопки редактирование списка стандартных высот;
* void saveHeights() – функция, обрабатывающая нажатие кнопки сохранить список стандартных высот;
* void saveMatrix() – функция, обрабатывающая нажатие кнопки сохранить матрицу превышений;
* void calculateMatrix() – функция, обрабатывающая нажатие кнопки расчет матрицы превышений;

Класс Matrix – с помощью этого класса происходит реализация расчета матрицы превышений и создание матрицы корректировок.

* void CreateFile (MRect rect, MString filename, double st = STEP, int y = 0, int x = 0) – функция создания матрицы на жестком диске;
* void Create (MRect rect, double st = STEP, int y = 0, int x = 0) – функция создания матрицы в памяти;
* void Destroy() – функция обработки процедуры удаления матрицы;
* void Calculate() – функция расчёта матрицы превышений;
* void Save (MString filename) – функция сохранения в файл;
* bool Load (MString filename) – функция загрузки матрицы в память;
* bool Open (MString filename) – функция открытия матрицы из файла на жестком диске;
* void Clear() – функция заполняющая матрицу нулями;
* MPoint GetPoint (int i, int j) – функция, которая возвращает точку, соответствующую i-ой строке и j-ому столбцу матрицы;
* void GetPos (MPoint point, int &i, int &j) – функция, определяющая позицию клетки матрицы, которая соответствует точке на карте;
* int GetX() – функция, определяющая количество столбцов матрицы;
* int GetY() – функция, определяющая количество строк матрицы;
* double GetStep() – функция, определяющая шаг матрицы;
* short GetHeight (MPoint point) – функция, которая возвращает высоту в точке;
* short GetHeight (int i, int j) – функция, которая возвращает высоту в клетке;
* void SetHeight (MPoint point, short height) – функция, устанавливающая высоту в клетке, соответствующей точке;
* void SetHeight (int i, int j, short height) – функция устанавливающая высоту в клетке;
* void ReadBmhHeader (FILE \*file) – функция, которая читает заголовок из файла BMH;
* void WriteBmhHeader (FILE \*file) – функция, которая записывает заголовок в файл BMH;
* void Fill (MPtArray \*ar) – функция, которая заполняет локальную матрицу обрабатываемого топографического объекта;
* int GetNumFilledPoints() – функция, определяющая количество заполненных клеток;
* void AddHeight (MTopobject \*tpo, short height) – функция, добавляющая высоты топографического объекта в матрицу;
* void AddLineObject (MTopobject \*tpo, short height) – функция, добавляющая высоты линейного объекта в матрицу;
* void AddPloObject (MTopobject \*tpo, short height) – функция добавляющая высоты площадного объекта в матрицу;
* void Expand (int i, int j, int state) – функция распространяющая заливку из клетки до границ объекта.

Последовательность вызова классов во времени представлена на диаграмме последовательностей (см. черт. РТДП 5.000.005).

## Описание используемых функции библиотеки картографии

Класс MKartaInterface предоставляет набор функций необходимых для работы с ядром библиотеки картографии.

Основные функции класса MKartaInterface:

* bool LoadMap (MString map\_name);

загружает карту (upt-файл) указанную в параметре map\_name.

* bool SetKlassifikator (MString kls\_name);

загружает классификатор (kls-файл) указанный в параметре kls\_name.

* void SetWidget (QWidget\* widget);

устанавливает widget на который должна отрисовываться карта.

* void DrawMap (bool toScreen = true);

отрисовывает карту, параметр toScreen определяет, будет ли карта отрисовываться на экран (true), либо в pixmap (false).

* MPoint LogicToGeo (QPoint& lPoint, bool bInMapCoo = false);
* QPoint GeoToLogic (MPoint& gPoint);

пересчитывают точку из логических (экранных) координат в географические и обратно.

* MRect LogicToGeo (QRect& lrect, bool bInMapCoo = false);
* QRect GeoToLogic (MRect& lrect);

пересчитывают прямоугольник из логических (экранных) координат в географические и обратно.

* MPoint GeoToGauss (MPoint& pt, double meredian = 1000);
* MPoint GaussToGeo (MPoint& pt, double meredian = 1000);

пересчитывают точку из географических координат в зонную систему координат Гаусса-Крюгера и обратно.

* MPoint GeoToGaussStandart (MPoint& pt, double meredian = 1000);
* MPoint GaussToGeoStandart (MPoint& pt, double meredian = 1000);

пересчитывают точку из географических координат в стандартную систему координат Гаусса-Крюгера и обратно.

* MPoint GeoToGaussLocal (MPoint& pt);
* MPoint GaussLocalToGeo (MPoint& pt);

пересчитывают точку из географических координат в местную систему координат Гаусса-Крюгера и обратно.

* void ZoomTo (MRect& rect);

зуммирует карту до заданного в параметре rect прямоугольника охвата.

* void ScrollByGeo (double x, double y);

скроллирует карту на смещение заданное в координатах карты.

* void ScrollByLogic (int x, int y);

скроллирует карту на смещение заданное в координатах окна отрисовки.

* bool SetMatrixFile (MString path);

устанавливает путь к файлу матрицы высот.

* int GetHight (MPoint& pt);

возвращает высоту рельефа в точке, заданной параметром pt.

* int GetHightObjects (MPoint& pt);

возвращает высоту объекта местности относительно рельефа в точке, заданной параметром pt.

Иерархия основных классов библиотеки картографии представлена на плакате (см. плакат РТДП 5.000.004).

# Разработка программных модулей

Среди используемых классов программы можно выделить Matrix, как основной для функций внутренней обработки и MapView – основной для функций отображения. Далее опишем алгоритмы ключевых процедур и функций этих классов.

## Алгоритмы ключевых процедур и функций класса MapView

void MapView:drawMap()

отрисовка карты производится, если открыты карта и классификатор (для определения выполнения этих условий в классе есть логические переменные mapOpen && klsOpen). Если на момент отрисовки карты открыта матрица превышений, нужно ее отобразить (функция updateHeights())

void MapView:updateHeights()

перерисовка (отображение) матриц превышений и корректировки на текущем отображаемом фрагменте карты производится, только если открыта карта или матрицы превышений и корректировки (используем логические переменные mapOpen и matrixOpen). С помощью класса библиотеки картографии MRect получаем границы карты, которые соответствуют видимой области (MRect bounds = karta->GetZoombounds()), далее вычисляем индексы ячеек матрицы, соответствующие углам карты (matrix->GetPos (MPoint(bounds.left, bounds.bottom), i1, j1); matrix->GetPos (MPoint(bounds.right, bounds.top), i2, j2);). Рассчитываем шаг матрицы double step. В logx и logy – содержится количество пикселей, которым соответствует шаг матрицы (double logx = width() / karta->GetZoombounds().Width() \* step; double logy = height() / karta->GetZoombounds().Height() \* step;). Матрицу превышений и матрицу корректировок отрисовываем таким образом: если масштаб крупный (шаг матрицы больше 5 пикселей), то рисуем прямоугольник, в противном случае ставим точку.

void MapView:mousePressEvent (QMouseEvent \*e)

корректировка в точке матрицы осуществляется только, если была нажата правая клавиша мыши (e->button() == Qt: RightButton) и открыты карта и матрица корректировки (используем логические переменные mapOpen && correctionMatrixOpen). С помощью функций класса MKartaInterface LogicToGeo и GeoToGauss производим преобразование логических координат мыши в географические координаты (MPoint point = karta->LogicToGeo(pos); point = karta->GeoToGauss(point);). Далее проверяем, если точка щелчка мыши находится в области матрицы корректировок, то вызывается диалог, в поле ввода которого вводим значение корректировки высоты. Значение высоты записываем в матрицу корректировок (correctionMatrix->SetHeight (point, height)) и вызываем перерисовку (отображение) матрицы корректировки (updateHeights()).

void MapView:zoom (int left, int top, int right, int bottom)

точки противоположных углов прямоугольника переводим из логических координат мыши в координаты карты

(QPoint first = QPoint (left, top);

QPoint second = QPoint (right, bottom);

MPoint p1 = karta->LogicToGeo(first);

MPoint p2 = karta->LogicToGeo(second);)

Формируем прямоугольник с помощью класса библиотеки картографии MRect (MRect rect = MRect (p1, p2)) и вызываем функцию ZoomTo (в качестве параметра передаем в нее сформированный прямоугольник rect). Для перерисовки карты вызываем функцию drawMap().

## Алгоритмы ключевых процедур и функций класса Matrix

void Matrix: CreateFile (MRect rect, MString filename, double st, int y, int x)

создание матрицы на жестком диске осуществляется, если была удалена предыдущая матрица (Destroy()), далее увеличиваем границы матрицы так, чтобы они были кратны шагу (NormalizeRect(rect)), записываем заголовок в файл ВМН (WriteBmhHeader(mem)) и записываем 0 в конец файла, создавая таким образом файл нужного размера, заполненный нулями.

fseek (mem, 48 + cx \* cy \* sizeof(short) – 1, SEEK\_SET);

char c = 0;

fwrite (&c, 1, 1, mem);

short Matrix: GetHeight (MTopobject \*tpo)

высота топографического объекта добавляется в матрицу из UTP-файла (MString str = tpo->GetDataPole(1)), если в данных объекта отсутствует значение высоты, оно берется из списка стандартных высот (heights [tpo->m\_kkod])

void Matrix: Save (MString filename)

сохранение в файл матрицы осуществляется в случае, если файл успешно создан на диске: FILE \*file = fopen (filename, «wb»). Для сохранения матрицы сначала записываем заголовок файла ВМН (WriteBmhHeader(mem)) и сохраняем матрицу или в оперативную память (если memMode == INMEMORY) или в файл на жесткий диск (если memMode == INFILE).

void Matrix: Calculate()

при расчете матрицы первоначально создаем матрицу с размерами равными размерам карты (Create (pKarta->m\_geobounds)), далее подсчитываем количество точек у всех топографических объектов, участвующих в расчете (это нужно, чтобы выводить полосу прогресса). Для формирования матрицы превышений, перебирая каждый объект каждого слоя карты, получаем высоту объекта. Если значение высоты не равно нулю, добавляем значение в формируемую матрицу превышений (с помощью функции AddHeight). При этом в матрице заполняются все клетки, покрываемые площадью рассматриваемого объекта.

void Matrix: AddHeight (MTopobject \*tpo, short height)

для добавления высоты топографического объекта в матрицу первоначально необходимо проверить тип топографического объекта (tpo->m\_pZnak->GetYaz()->GetType()). Если объект площадной, то выполняется добавление высоты площадного объекта в матрицу (AddPloObject (tpo, height)), если объект линейный, то выполняется добавление высоты линейного объекта в матрицу (AddLineObject (tpo, height)), если объект точечный, то сравниваем текущую высоту с высотой в ячейке, если текущая высота больше (height > GetHeight(point), то устанавливаем в ячейке текущую высоту (SetHeight (point, height))

void Matrix: AddLineObject (MTopobject \*tpo, short height)

добавление высоты линейного топографического объекта в матрицу осуществляется следующим образом: в цикле по сегментам линейного объекта проверяем покрывает ли ячейка матрицы сегмент, если покрывает, то устанавливаем в ячейке текущую высоту (SetHeight (point, height)), если не покрывает, то рассчитываем количество ячеек покрывающих данный сегмент, и в цикле для каждой ячейки устанавливаем текущую высоту топографического объекта.

void Matrix: AddPloObject (MTopobject \*tpo, short height)

добавление высоты площадного топографического объекта в матрицу осуществляется следующим образом: сначала создается локальная матрица топографического объекта (по размеру охватывающего прямоугольника объекта). Затем в локальной матрице площадного объекта заполняются ячейки, покрывающие контур объекта – используется функция AddLineObject (tpo, BORDER). Затем заполняются остальные ячейки локальной матрицы с использованием функции Fill (MPtArray \*ar). На последнем этапе в цикле по всем ячейкам сформированной локальной матрицы все «внутренние» (не-OUTSIDE) ячейки заполняются значениями высоты площадного объекта, но только в том случаи если текущее значение высоты больше значения высоты в ячейке

if (height > GetHeight(point))

SetHeight (point, height);

void Fill (MPtArray \*ar)

заполнение локальной матрицы обрабатываемого топографического объекта продолжается до тех пор, пока не будут заполнены все точки (GetNumFilledPoints() < numPoints). Первоначально в цикле do while выбираем случайную ячейку со значением высоты EMPTY (не заполненная) и вычисляем соответствующую точку объекта (MPoint pt = GetPoint (i, j)). Затем распространяем вокруг этой клетки значение высоты объекта с помощью функции Expand(), учитывая при этом, является точка внутренней (INSIDE) или наружной (OUTSIDE) по отношению к объекту.

if (PointInRegion (pt, ar)) Expand (i, j, INSIDE);

else Expand (i, j, OUTSIDE);

void Expand (int i, int j, int state)

распространение заливки из клетки до границ объекта происходит следующим образом: сначала устанавливаем значение в клетке (SetHeight (i, j, state)), затем в цикле заполняем соседние клетки, обход осуществляется по кругу, вначале по часовой стрелке, потом – против, после каждой итерации увеличиваем радиус круга. Если после обхода не заполнено ни одной точки, значит распространяться «заливке» уже некуда, выходим из цикла.

# 

# Программа и методика испытаний

## 

Разработанная программа должна функционировать с определенными характеристиками для заданных областей данных. Процесс, позволяющий получить программу, функционирующую с требуемыми характеристиками в заданной области входных данных, называется отладкой. В результате отладки программа должна соответствовать определенной фиксированной совокупности правил и показателей качества.

Основным методом отладки и обнаружения ошибок является тестирование – важнейший фактор, определяющий стоимость и длительность разработки комплекса программ. Затраты на тестирование составляют до 40% от общего времени на разработку программы.

Программы, как объекты тестирования, имеют ряд особенностей, которые отличают процесс тестирования технического средства от программного. Для сложных комплексов программ отсутствует полный и точный эталон для тестовых наборов. Для сложных программ тестирование проводится в объемах минимально необходимых для проверки программ в ограниченных пределах изменения параметров и условий функционирования.

Для проверки правильности работы программы используется проведение функционального и полного тестирования.

При функциональном тестировании должна обеспечиваться проверка каждого пункта меню, каждой операции, которую выполняет система. Такая проверка должна гарантировать работу всех элементов программы в автономной режиме.

При полном тестировании моделируются все возможные действия пользователя при работе с программой.

Сначала тестирование было проведено разработчиком программы, затем программный продукт был дан на тестирование пользователям. При этом на первом этапе выявлялись логические ошибки в основном связанные с последовательностью и временем появления диалогов на экране, которые затем были устранены.

Тестирование проводилось также по следующему алгоритму: тестировщик проходит по всем этапам программы, начиная от главного окна. При этом он бессистемно нажимает все клавиши на клавиатуре и мыши. Тест показал, программа реагирует на нажатие только тех клавиш, которые имеют смысловое значение в программном модуле.

Для проверки функциональности программы был разработан представленный ниже тест, включающий в себя последовательно описанные действия оператора, направленные на проверку определенных функций программного модуля.

## Подготовка к тестированию

Тестирование программного модуля обработки и отображения картографической информации выполняется в среде функционирования программного модуля.

Подготовка средств

Необходимо обеспечить соответствие среды функционирования программного модуля обработки и отображения картографической информации требованиям п. 6.1 настоящего документа.

В среде функционирования программного модуля требуется:

* установить программное обеспечение модуля, следуя инструкциям п. 6.2 настоящего документа;
* записать в отдельный каталог файлы тестового примера (файл карты m3404.utp, файл классификатора 500–200-HOT\_heights.kls, файл стандартных высот heights.txt и файл матрицы высот matrix.bmh).

Файлы тестового примера предоставляет исполнитель.

## Ожидаемые результаты

В результате проведения тестирования ожидаются следующие результаты:

* после открытия карты и классификатора, в главном окне программного модуля ожидается отображение цифровой карты местности (ЦКМ);
* после использования инструмента зуммирования ожидается отображение в картографическом окне выделенного фрагмента ЦКМ, увеличенного до максимально возможного размера, который позволяет отобразить картографическое окно. Если размеры картографического окна и увеличиваемого фрагмента ЦКМ не симметричны, то отображаемый фрагмент дополняется участками ЦКМ, прилегающими к выделенному фрагменту, для полного заполнения картографического окна в вертикальном или горизонтальном направлении;
* после использования инструмента скроллирования ожидается отображение в картографическом окне необходимого фрагмента ЦКМ;
* после использования инструмента «предыдущий вид карты» ожидается отображение в картографическом окне предыдущего фрагмента ЦКМ;
* после использования инструмента «отобразить всю карту» ожидается отображение в картографическом окне всей ЦКМ;
* после завершения работы программного модуля Matrix, в директории указанной тестировщиком, должны находиться следующие сохраненные файлы: файл матрицы превышений matrix\_m3404.bmh2 и файл матрицы корректировок correction.bmh2.

## Порядок выполнения теста

Для выполнения тестирования программного модуля обработки и отображения картографической информации на цифровой карте местности следует последовательно выполнить следующие действия:

* запустить программный модуль Matrix (см. п. 6.3. настоящего документа);
* открыть карту m3404.utp и классификатор 500–200-HOT\_heights.kls (см. п. 6.5. настоящего документа);
* создать файл стандартных высот heights.txt (см. п. 6.5. настоящего документа);
* если необходимо, внести изменения в список стандартных высот в диалоге «Редактирование высот» (см. п. 6.5. настоящего документа);
* сохранить файл стандартных высот heights.txt (см. п. 6.5. настоящего документа);
* произвести расчет матрицы превышений (см. п. 6.5. настоящего документа);
* сохранить матрицу превышений matrix\_m3404.bmh2 (см. п. 6.5. настоящего документа);
* создать матрицу корректировок correction.bmh2 (см. п. 6.5. настоящего документа);
* если необходимо, внести изменения в матрицу корректировок   
  (см. п. 6.5. настоящего документа);
* проверить функции инструментов зуммирования, скроллирования, «предыдущий вид» и «отображение всей карты» (см. п. 6.5. настоящего документа);
* завершить работу программного модуля Matrix (см. п. 6.5 настоящего документа).

Чтобы убедиться в правильности расчета матрицы превышений необходимо закрыть программный модуль Matrix и затем открыть его снова. При этом выполнить все действия повторно, но уже не рассчитывать матрицу превышений, а открыть сохраненную ранее. При движении мыши в картографическом окне, если матрица рассчитана правильно, в строке состояния должны отображаться значения матрицы превышения и матрицы корректировки.

# Руководство пользователя

## 

Для оптимальной работы с программным модулем обработки и отображения картографической информации на цифровой карте местности персональный компьютер пользователя должен обладать следующими характеристиками:

* оперативная память – 128 Мб и более;
* жесткий диск общим объемом не менее 20 Гб со свободными 500 Мб – 1 Гб;
* монитор должен поддерживать разрешение 1024\*768 пикселей и цветопередачу True Color;
* дисковод для CD-дисков;
* оснащение мышью;
* установленная операционная система Windows 2000;

## Инсталляция

Все файлы необходимые для работы программного модуля организованы в корневом каталоге MATRIX:

* matrix.exe – исполняемый файл;
* gt-mt333.dll – библиотечный файл;
* MSVCRTD.DLL – библиотечный файл;
* MFC42D.DLL – библиотечный файл;
* MFCO42D.DLL – библиотечный файл;
* MSVCIRTD.DLL – библиотечный файл;
* MSVCP60D.DLL – библиотечный файл.

Для установки модуля на персональный компьютер пользователя достаточно скопировать на жесткий диск каталог MATRIX с любого машинного носителя.

Для начала работы следует запустить программу matrix.exe.

## Запуск программы

Для запуска программы Matrix необходимо щелкнуть пиктограмму, соответствующую программе. После запуска на экране монитора появляется главное окно программы.

В левой части окна расположены кнопки вызова функций. Функции вызываются щелчком мыши по выбранной кнопке:

- открыть карту;

- открыть папку;

- открыть классификатор;

- создать перечень стандартных высот;

- открыть перечень стандартных высот;

– сохранение созданного перечня стандартных высот;

– редактирование перечня стандартных высот;

- открыть матрицу превышений;

– сохранение матрицы превышений;

- расчет матрицы превышений;

- создать матрицу корректировок;

- открыть матрицу корректировок.

В верхней части окна расположены кнопки вызова функций управления картой. Функции вызываются щелчком мыши по выбранной кнопке.

– отображение всей карты;

- возвращение к предыдущему экрану отображения;

– зуммирование (изменение отображения фрагмента карты);

– прокрутка.

## Управление картографическим окном

Цифровая карта местности хранится на жестком диске АРМ в файлах четырех типов:

* файлы объектов местности с расширением UTP;
* файл-классификатор объектов с расширением KLS;
* файл матрицы высот с расширением BMH;
* файл превышения высот с расширением BMH2.

Файлы объектов местности содержат пространственные, количественные, качественные и описательные характеристики объектов. Один файл является базовым, т.к. покрывает весь интересующий регион и содержит минимальный набор объектов. Остальные файлы объектов соответствуют листам карт масштаба 1:200000 или 1:500000.

Файл классификатора объектов карты содержит данные об их представлении на экране монитора в зависимости от масштаба отображения. В классификатор дополнительно включено описание представления объектов оперативной информации на фоне объектов карты.

Файл матрицы высот содержит данные о возвышении земной (водной) поверхности относительно уровня Балтийского моря в каждой точке местности.

Файл превышения высот содержит данные о возвышении верхней поверхности сооружений, растительного покрова относительно земной поверхности в каждой точке местности.

Объекты карты распределены по слоям (гидрография, границы и т.д.). Отображение карты производится последовательно слой за слоем в соответствии с классификатором.

Для улучшения зрительного восприятия слоя и ускорения процесса его отображения применяется упрощенная отрисовка.

Степень детализации отображаемой карты изменяется установкой минимального размера отображаемого объекта. Размер задается в единицах измерения экранной системы координат, т.е. в пикселях.

## Начало работы

Перед началом работы необходимо назначить рабочую папку. Для этого требуется нажать кнопку (открыть папку), после чего на экране отобразится диалоговое окно «Выберите папку».

В строке Directory задать название рабочей папки и нажать клавишу (ОК).

Первоначально для загрузки карты необходимо нажать кнопку (открыть карту). Отобразится диалоговое окно «Открыть карту».

Выбрать базовый файл карты и нажать кнопку (Open).

Далее необходимо нажать кнопку (открыть классификатор). Отобразится диалоговое окно «Открыть классификатор».

Выбрать файл классификатора и нажать кнопку (Open).

Далее для расчета матрицы высот необходимо создать стандартный перечень высот. Для этого необходимо нажать на кнопку (создать перечень стандартных высот), по умолчанию создаваемому файлу со списком высот задается имя heights.txt, а после нажать кнопку (редактирование перечня стандартных высот).

В нем из классификатора в список высот с помощью кнопки выбираются необходимые типы объекты. Для каждого типа объекта (пункта из списка) задается с клавиатуры своя высота.

После того, как были заданы высоты для всех выбранных типов объектов, необходимо нажать кнопку (выход). Далее необходимо сохранить файл со стандартными высотами, для этого нажать кнопку (сохранить) (см. рис. 6.1). На экране появится диалоговое окно «Сохранить список высот».

В строке «Имя файла» задать название файла и нажать кнопку (сохранить).

Если требуется открыть уже сохраненный файл со стандартными высотами, необходимо нажать кнопку (открыть) и в диалоговом окне «Открыть список высот» выбрать ранее сохраненный файл и нажать кнопку (открыть).

Для вызова функции расчета матрицы превышений необходимо нажать кнопку (расчет матрицы превышений). После чего в нижнем правом углу главного окна появится полоса прогресса, по достижении 100% на экране должно отобразиться окно.

Далее необходимо сохранить файл матрицы превышений высот, для этого необходимо нажать кнопку (сохранить). На экране появится диалоговое окно «Сохранить матрицу превышений».

В строке «Имя файла» задать название файла и нажать кнопку (сохранить).

Расчет матрицы превышений происходит с некоторыми погрешностями, которые связаны с шагом расчета в матрице. И для того, что бы скорректировать в отдельных точках значения высот можно создать матрицу корректировок. Для этого необходимо нажать кнопку (создать матрицу корректировок). На экране отобразится диалоговое окно.

По нажатию клавиши (да) матрица корректировок будет создана, по нажатию на клавишу (нет) – не будет.

Для загрузки ранее сохраненного файла матрицы корректировок необходимо нажать на кнопку (открыть). Откроется диалоговое окно «Открыть матрицу корректировок».

В процессе работы может появиться необходимость рассчитать матрицу превышений для нескольких листов карты одновременно. Для этого необходимо нажать кнопку (открыть папку). Выбрать папку, в которой уже лежат несколько карт одного масштаба в формате UTP. Затем нажать кнопку (открыть классификатор) и выбрать классификатор нужного масштаба. Для продолжения работы необходимо нажать кнопку (редактирование перечня стандартных высот), откроется диалоговое окно «Стандартные высоты».

В нем из классификатора в список высот с помощью кнопки выбираются необходимые типы объектов. Для каждого типа объекта (пункта из списка) задается с клавиатуры своя высота.

После того, как были заданы высоты для всех выбранных типов объектов, необходимо нажать кнопку (выход). Далее необходимо сохранить файл со стандартными высотами, для этого нажать кнопку (сохранить) (см. рис. 6.1). На экране появится диалоговое окно «Сохранить список высот». В строке «Имя файла» задать название файла и нажать кнопку (сохранить). Для вызова функции расчета матрицы превышений необходимо нажать кнопку (расчет матрицы превышений).

Для завершения работы диалога необходимо нажать кнопку (OК).

После создания матрицы превышений можно осуществить корректировку матрицы превышений. Далее следует работа с картой, для этого предусмотрено несколько инструментов.

Для отображения в картографическом окне всей карты необходимо нажать кнопку (исходный размер карты) (см. рис. 6.1).

Для выполнения операции прокрутки карты необходимо нажать кнопку (прокрутка) и, нажав и удерживая в картографическом окне левую кнопку мыши, можно переместить изображение карты в нужную область.

Для выполнения операции зуммирования необходимо нажать кнопку (увеличить). Далее, нажав и удерживая левую кнопку мыши, необходимо выделить часть карты, которую необходимо увеличить. После отпускания кнопки мыши в картографическом окне отобразится выделенный участок карты.

Для возвращение назад необходимо нажать кнопку (предыдущий вид).

Последовательность действий при обработке и отображении картографической информации представлена на плакате.

# Технико-экономическое обоснование

## 

Целью диплома является написание программного модуля обработки и отображения картографической информации на цифровой карте местности. Заказчиком данного программного модуля является УП «НИИСА». Это программное средство должно выполнять следующие функции (задачи):

* загрузка, отображение цифровой карты местности (ЦКМ), прокрутка и зуммирование изображения загруженной цифровой карты местности (с использованием функций существующей библиотеки картографического обеспечения);
* создание, редактирование и сохранение списка стандартных высот для объектов цифровой карты местности (по видам объектов, согласно классификатора условных знаков);
* расчет матрицы превышений на фоне цифровой карты местности (с учетом списка стандартных высот, соответствующих данному участку местности);
* создание и редактирование матрицы корректировок.

Программные средства (ПС) вычислительной техники (ВТ) являются материальными объектами специфической интеллектуальной деятельности специалистов, состоящими из программных документально оформленных проектов, реализующих свои потребительские свойства и качества в составе функционирующих вычислительных систем или систем обработки данных. По стоимости и срокам службы ПС относятся к основным производственным фондам предприятия.

Широкое применение ВТ требует постоянного обновления и совершенствования ПС. Выбор эффективных проектов ПС связан с их экономической оценкой и расчетом экономического эффекта.

Новый программный продукт должен быть не только технически совершенным, но и экономически выгодным. На основе экономической оценки нового программного продукта принимается решение об инвестициях в данный проект. Отсутствие экономического обоснования и расчетов приводит к экономическим ошибкам в проектировании, созданию неэффективных проектов.

В результате разработки и применения программного модуля обработки и отображения картографической информации на цифровой карте местности экономический эффект достигается за счет экономии трудовых, материальных и финансовых ресурсов по сравнению с базовым вариантом, программного модуля. Если программное средство – товар, то возможен расчет экономического эффекта, который выполнен на 29.05.06.

## Смета затрат и цена программного обеспечения

Определение объема и трудоемкости программного обеспечения (ПО)

Объем ПО. Базой для расчета плановой сметы затрат на разработку ПО является объем ПО.

Общий объем (V0) программного продукта определяется исходя из количества и объема функций, реализуемых программой

, (7.1)



где: Vi – объем отдельной функции ПО;

n – общее число функций.

Расчет общего объема функций по каталогу представлен в таблице 7.1.

Единицы измерения объема ПО. Оценивание объема программного продукта связано с выбором наиболее подходящие единицы измерения размера продукта. В данном дипломном проекте, в качестве единицы измерения объема ПО, используется количество строк исходного кода (Lines Of Code, LOC).

Строка исходного кода (LOC) является универсальной метрикой, так как может применяться при создании любых программных продуктов.

Расчет объема программного продукта (количества строк исходного кода) предполагает определение типа программного обеспечения, всестороннее техническое обоснование функций ПО и определение объема каждой функции. На стадии технико-экономического обоснования проекта невозможно рассчитать точный объем функций. Могут быть получены только ориентировочные (прогнозные) оценки на основе имеющихся фактических данных по аналогичным проектам, выполненным ранее, или путем применения действующих нормативов. На основании информации о функциях разрабатываемого ПО по каталогу функций определяется объем функций и общий объем ПО, который уточняется (корректируется). Рассчитывается уточненный объем ПО (Vy)

, (7.2)



где: Vyi – уточненный объем отдельной функции ПО;

Расчет уточненного объема ПО представлен в таблице 7.1.

Таблица 7.1. Перечень и объем функций программного модуля

|  |  |  |  |
| --- | --- | --- | --- |
| № функции | Наименование (содержание) | Объем функции (LOC) | |
| по каталогу Vi | уточненный Vyi |
| 1 | 2 | 3 | 4 |
| 101 | Организация ввода информации | 150 | 75 |
| 109 | Организация ввода / вывода информации в интерактивном режиме | 320 | 250 |
| 301 | Формирование последовательного файла | 290 | 120 |
| 305 | Обработка файлов | 720 | 622 |
| 309 | Формирование файла | 1020 | 571 |
| 703 | Расчет показателей | 460 | 295 |
| 707 | Графический вывод результатов | 480 | 325 |
|  | Итого: | 3440 | 2258 |

В связи с использованием более совершенных средств автоматизации общие объемы функций были уменьшены. Уточненный объем ПО () составил 2258 LOC, общий объем функций по каталогу (V0) 3440 LOC.



За общий объем (V0) программного продукта принимаем значение уточненного объема функций который равен 2258 LOC.

Трудоемкость разработки ПО. По уточненному объему ПО и нормативам затрат труда в расчете на единицу объема определяется нормативная и общая трудоемкость разработки ПО.

Нормативная трудоемкость разработки ПО. На основании принятого к расчету объема () и категории сложности определяется нормативная трудоемкость ПО (), которая уточняется с учетом сложности, новизны проекта и степени использования стандартных модулей при разработке. В данном дипломном проекте =2258, следовательно =61.



Общая трудоемкость разработки ПО. Нормативная трудоемкость () служит основой для определения общей трудоемкости (To), расчет которой осуществляется в зависимости от размера проекта.



Общая трудоемкость в данном проекте рассчитывается по формуле:

, (7.3)



где: – коэффициент, учитывающий сложность ПО;



– поправочный коэффициент, учитывающий степень использования при разработке стандартных модулей;



– коэффициент, учитывающий степень новизны ПО.



Категория сложности ПО. Все ПО принято подразделять на три категории сложности, в зависимости от наличия (отсутствия) следующих характеристик:

* высокий уровень языкового интерфейса с пользователем;
* режим работы в реальном времени;
* управление удаленными объектами;
* машинная графика;
* существенное распараллеливание вычислений;
* переносимость ПО;
* и другие.

Влияние фактора сложности на трудоемкость учитывается умножением нормативной трудоемкости на соответствующий коэффициент сложности. Данный программный модуль обладает 2 категорией сложности.

Коэффициент сложности (). Посредством коэффициента сложности учитываются дополнительные затраты труда, связанные со сложностью разрабатываемого программного продукта. Коэффициент сложности рассчитывается по формуле:



, (7.4)



где: – коэффициент, соответствующий степени повышения сложности ПО.



В разрабатываемом дипломном проекте , за счет наличия у программного модуля одновременно двух характеристик:



* машинная графика;
* обеспечение переносимости ПО.

принимаем =0,12.



Коэффициент, учитывающий степень использования при разработке ПО стандартных модулей (). Степень использования в разрабатываемом ПО стандартных модулей определяется их удельным весом в общем объеме проектируемого продукта. В данном дипломном проекте степень охвата реализуемых функций разрабатываемого ПО стандартными модулями, типовыми программами и ПО до 20%, следовательно =0,9.



Коэффициент новизны разрабатываемого ПО (). Сравнение характеристик разрабатываемого ПО с имеющимися аналогами позволяет определить экспертным путем степень его новизны. Разрабатываемый программный модуль является принципиально новым ПО, не имеющим доступных аналогов, в соответствии с вышеизложенным =1,0.



61\*1,12\*0,9\*1=62 (чел./дн.)



Численность исполнителей и срок разработки ПО

На основе общей трудоемкости определяется плановое число разработчиков и плановые сроки, необходимые для реализации проекта в целом . При этом решаются следующие задачи:



* расчет числа исполнителей при заданных сроках разработки проекта;
* определение сроков разработки проекта при заданной численности исполнителей.

Численность исполнителей проекта рассчитывается по формуле:



(7.5)



где: - эффективный фонд времени работы одного работника в течение года (дн.);



- общая трудоемкость разработки проекта (чел./дн.);



- срок разработки проекта (лет).



Срок разработки проекта () определяется по формуле:



(7.6)



В данном дипломном проекте срок разработки проекта установлен 3 месяца, что составляет 0,25 года, следовательно =0,25 года.



Эффективный фонд времени работы одного работника () рассчитывается по формуле:



, (7.7)



где: - количество дней в году;



- количество праздничных дней в году;



- количество выходных дней в году;



- количество дней отпуска.



В разрабатываемом дипломном проекте будем принимать значения:

=32 дня.



Эффективный фонд времени работы одного работника () рассчитывается по формуле 7



(дн.).



Численность исполнителей проекта рассчитывается по формуле 5



=1 (чел.).



Исходные данные для последующих расчетов, а также результаты расчетов объема и трудоемкости разработки программного обеспечения, численность исполнителей и срок разработки сведены в таблицу 7.2.

Таблица 7.2. Исходные данные и результаты расчетов

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование показателей | Буквенное обозначение | Единицы измерения | Кол-во |
| 1 | 2 | 3 | 4 |
| Коэффициент новизны | Кн |  | 1,0 |
| Категория сложности |  |  | 2 |
| Дополнительный коэффициент сложности | Кi |  | 0,12 |
| Поправочный коэффициент, учитывающий использование типовых программ | Кт |  | 0,9 |
| Установленная плановая продолжительность разработки | Тр | год | 0,25 |
| Годовой эффективный фонд времени | Фэф | дней | 221 |
| Продолжительность рабочего дня | Тч | ч | 8,0 |
| Месячная тарифная ставка 1-го разряда | Тм1 | руб. | 100000 |
| Коэффициент премирования | К |  | 1,5 |
| Норматив дополнительной заработной платы | Нд | % | 20,0 |
| Ставка отчислений в фонд социальной защиты населения | Нсз | % | 35,0 |
| Ставка чрезвычайного налога | Нне | % | 4,0 |
| Ставка отчислений в фонд занятости | Ннз | % | 1,0 |
| Норматив прочих затрат | Нпз | % | 20,0 |
| Норматив накладных расходов | Нрн | % | 80,0 |
| Уровень рентабельности ПО | Урп | % | 20,0 |
| Норматив отчислений в местный и республиканский бюджеты | Нмр | % | 3,0 |
| НДС | Ндс | % | 18,0 |
| Норматив расходов на освоение | Но | % | 10,0 |
| Норматив расходов на сопровождение | Нс | % | 20,0 |
| Норматив налога на прибыль | Ннп | % | 24,0 |

Основной статьей расходов на создание ПО является заработная плата разработчиков (исполнителей) проекта, в число которых принято включать инженеров-программистов, участвующих в написании кода, руководителей проекта, системных архитекторов и других. Заработная плата руководителей организации и работников вспомогательных служб (инфраструктуры) учитывается в накладных расходах.

Расчет основной заработной платы исполнителей.

В разработке данного программного модуля задействован следующий исполнитель:

* инженер (тарифный разряд – 12; тарифный коэффициент – 2,84; продолжительность участия в разработке – 0,25 года (3 мес.)).

Общая трудоемкость, плановая численность работников и плановые сроки разработки ПО являются базой для расчета основной заработной платы разработчиков проекта.

Месячная тарифная ставка исполнителя (Тм) определяется путем умножения действующей месячной тарифной ставки 1-го разряда (Тм1) на тарифный коэффициент (Тк), соответствующий установленному тарифному разряду:

, (7.8)



Месячная тарифная ставка 1-го разряда (Тм1) в УП НИИСА на 01.03.2006 года составляет 100 000 рублей.

Тарифный коэффициент (Тк) – 2,84.

=284 000 рублей



Часовая тарифная ставка (Тч) рассчитывается путем деления месячной тарифной ставки на установленную при 40-часовой недельной норме рабочего времени расчетную среднемесячную норму рабочего времени в часах (Фр):



, (7.9)



где: - по УП НИИСА в 2006 году составляет 168,3 часа;



=284 000 рублей



=284 000 / 168,3 = 1 688 (руб.).



Основная заработная плата исполнителя на конкретное ПО рассчитывается по формуле:

(7.10)



где: n – количество исполнителей, занятых разработкой конкретного ПО;

Tчi – часовая тарифная ставка исполнителя (руб.);

Фп – эффективный фонд рабочего времени исполнителя (дней);

ТЧ – количество часов работы в день (ч);

К – коэффициент премирования.

=1\*1 688\*62\*8\*1,5=1 255 872 (руб.)



Дополнительная заработная плата на конкретное ПО (Зд) включает выплаты, предусмотренные законодательством о труде (оплата отпусков, льготных часов, времени выполнения государственных обязанностей и других выплат, не связанных с основной деятельностью исполнителей), и определяется по нормативу в процентах к основной заработной плате:

(7.11)



где: Зд – дополнительная заработная плата исполнителя на конкретное ПО (руб.);

Нд – норматив дополнительной заработной платы.

=(1 255 872 \* 20%) / 100% = 251 174 (руб.)



Отчисления в фонд социальной защиты населения (Зсз) определяются в соответствии с действующими законодательными актами по нормативу в процентном отношении к фонду основной и дополнительной зарплаты исполнителей, определенной по нормативу, установленному в целом по организации:

(7.12)



где НСЗ – норматив отчислений в фонд социальной защиты населения (35%).

=((1 255 872 + 251 174) \* 35%) / 100% = 527 466 (руб.)



Налоги, рассчитываемые от фонда оплаты труда определяются в соответствии с действующими законодательными актами по нормативам в процентном отношении к сумме всей заработной платы, относимой на ПО (налог, уплачиваемый единым платежом, включая налог на ликвидацию последствий чернобыльской катастрофы и отчисления в фонд занятости (Не)):

(7.13)



где Нне – норматив налога, уплачиваемого единым платежом (4%).

=((1 255 872 + 251 174) \* 4%) / 100% = 60 282 (руб.)



Расходы по статье «Материалы» (М) определяются на основании сметы затрат, разрабатываемой на ПО с учетом действующих нормативов. По статье «Материалы» отражаются расходы на магнитную носители, перфокарты, бумагу, красящие ленты и другие материалы, необходимые для разработки ПО. Нормы расхода материалов в суммарном выражении (Нм) определяются в расчете на 100 строк исходного кода или по нормативу в процентах к фонду основной заработной платы разработчиков (Нмз), который устанавливается организацией (в УП НИИСА составляет 3%). Сумма затрат материалов рассчитывается по формуле:

(7.14)



= (1 255 872 \* 3%)/100% = 37 676 (руб.)



Расходы по статье «Машинное время» (Рм) включают оплату машинного времени, необходимого для разработки и отладки ПО, которое определяется по нормативам (в машино-часах) на 100 строк исходного кода (Нмв) машинного времени в зависимости от характера решаемых задач и типа ПЭВМ:

(7.15)



где: Цм – цена одного машино-часа (2 000 руб.);

Vо – общий объем ПО (строк исходного кода);

Нмв – норматив расхода машинного времени на отладку 100 строк исходного кода (10 машино-часов).

= 2000 \* (2258/100) \* 10 = 440 000 (руб.)



Расходы по статье «Прочие затраты» (Пз) на конкретное ПО включают затраты на приобретение и подготовку специальной научно-технической информации и специальной литературы. Определяются по нормативу, разрабатываемому в целом по научной организации, в процентах к основной заработной плате:

(7.16)



где: Нпз – норматив прочих затрат в целом по научной организации (20%).

=(1 255 872 \* 20%) / 100% = 251 174 (руб.)



Затраты по статье «Накладные расходы» (Рн), связанные с необходимостью содержания аппарата управления, вспомогательных хозяйств и опытных (экспериментальных) производств, а также с расходами на общехозяйственные нужды (Рн), относятся на конкретное ПО по нормативу (Нрн) в процентном отношении к основной заработной плате исполнителей. Норматив устанавливается в целом по научной организации:

(7.17)



где: Рн – накладные расходы на конкретную ПО (тыс. руб.);

Нрн – норматив накладных расходов в целом по научной организации (100%).

=1 255 872 рубля



Общая сумма расходов по всем статьям сметы (себестоимость) (Сп) на ПО рассчитывается по формуле:

(7.18)



= 4 079 516 рублей



Рентабельность и прибыль по создаваемому ПО определяются исходя из результатов анализа рыночных условий, переговоров с заказчиком (потребителем) и согласования с ним отпускной цены, включающей дополнительно налог на добавленную стоимость и отчисления в местный и республиканский бюджеты. В случае разработки ПО для использования внутри организации оценка программного продукта производится по действующим правилам и показателям внутреннего хозрасчета (по ценам, устанавливаемым для расчета за услуги между подразделениями). Прибыль рассчитывается по формуле:

(7.19)



где: Пс – прибыль от реализации ПО заказчику (тыс. руб.);

Урп – уровень рентабельности ПО (%);

Сп – себестоимость ПО (тыс. руб.).

= (4 079 516 \* 20%)/100% = 815 903 (руб.)



Прогнозируемая цена ПО без налогов (Цп):

(7.20)



=4 895 419 рублей



Отчисления и налоги в местный и республиканский бюджеты единым платежом (Омр):

(7.21)



где Нмр – норматив отчислений в местный и республиканский бюджеты (3%).

= (4 895 419 \* 3%) / (100% – 3%) = 151 405 (руб.)



Налог на добавленную стоимость (НДС)

(7.22)



где Ндс – норматив НДС (18%).

=((4 895 419 + 151 405) \* 18%) / 100% = 908 428 (руб.)



Прогнозируемая отпускная цена (Цо):

(7.23)



= 4 895 419 + 151 405 + 908 428 = 5 955 252 (руб.)



Затраты на освоение ПО (Ро). Организация – разработчик участвует в освоении ПО и несет соответствующие затраты, на которые составляется смета, оплачиваемая заказчиком по договору. Сметой предусматриваются не только затраты (основная и дополнительная зарплата, начисления на зарплату и т.д.), но и налоги, предусмотренные законодательством, и прибыль организации – разработчика. Для упрощения расчетов, до составления сметы, затраты на освоение определяются по нормативу (Но = 10%) от себестоимости ПО в расчете на 3 месяца и рассчитываются по формуле:

(7.24)



где Но – норматив расходов на освоение (10%).

= (4 079 516\*10%) / 100% = 407 952 (руб.)



Затраты на сопровождение ПО (Рс). Организация – разработчик осуществляет сопровождение ПО и несет соответствующие расходы, которые оплачиваются заказчиком в соответствии со сметой на освоение ПО. Для упрощения расчетов для составления сметы затраты на сопровождение определяются по установленному нормативу (Нс = 20%) от себестоимости ПО (в расчете на год) и рассчитываются по формуле:

(7.25)



где Нс – норматив расходов на сопровождение (20%).

= (4 079 516\*20%) / 100% = 815 903 (руб.)



Экономический эффект организации – разработчика. Заказчик оплачивает организации – разработчику всю сумму расходов по проекту, включая прибыль. После уплаты налогов из прибыли в распоряжении заказчика остается чистая прибыль от проекта.

Расчет прибыли, остающиеся в распоряжении предприятия, осуществляется по формуле:

(7.26)



где Ннп – норматив налога на прибыль (24%).

=815 903 – (815 903 \* 24%) /100% = 620 086 (руб.)



Чистая прибыль от реализации ПО (= 620 086 руб.) остается организации-разработчику и представляет собой экономический эффект от создания нового программного средства.



## Оценка экономической эффективности применения ПО у пользователя

Для определения экономического эффекта от использования нового ПС у потребителя необходимо сравнить расходы по всем основным статьями сметы затрат на эксплуатацию нового ПС (расходы на заработную плату с начислениями, затраты на расходные материалы, расходы на машинное время) с расходами по соответствующим статьям базового варианта. При этом за базовый вариант следует принимать аналогичное программное средство, используемое в действующей автоматизированной системе, или ручной вариант, если автоматизация отсутствует. При сравнении базового и нового вариантов ПО в качестве экономического эффекта будет выступать общая экономия всех видов ресурсов относительно базового варианта. Создание нового ПО окажется экономически целесообразным лишь в том случае, если все капитальные затраты окупятся за счет получаемой экономии в ближайшее время (2 лет).

Исходные данные для расчета приведены в таблице 7.3

Таблица 7.3. Исходные данные для расчета экономии ресурсов в связи с применением нового ПО

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Наименование показателей | Обозначения | Единицы измерения | Значение показателя | | Наименование источника информации |
| в базовом варианте | в новом варианте |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 1. Капитальные вложения, включая затраты пользователя на приобретение ПО | Кпр | руб. | - | 5 955 252 | Прогнозируемая отпускная цена |
| 1. Затраты на освоение ПО | Кос | руб. | - | 407 952 | 10% от себестоимости |
| 1. Затраты на сопровождение ПО | Кс | руб. | - | 815 903 | 20% от себестоимости |
| 1. Затраты на укомплектование ВТ техническими средствами в связи с внедрением нового ПО | Ктс | руб. | - | - | Сметы затрат на внедрение |
| 1. Затраты на пополнение оборотных средств в связи с эксплуатацией нового ПО | Коб | руб. | - | - | Сметы затрат на внедрение |
| 1. Среднемесячная ЗП одного программиста | Зсм | руб. | 400 000 | 400 000 | Расчетные данные пользователя |
| 1. Коэффициент начислений на зарплату | Кнз |  | 0,4 | 0,4 | Рассчитывается по данным пользователя |
| 1. Среднемесячное количество рабочих дней | Др | день | 21 | 21 | Принято для расчета |
| 1. Количество типовых задач, решаемых за год | Зт2 | задача | - | 100 | План пользователя |
| 1. Объем работ, выполняемых при решении одной задачи | А0 | 100 команд | - | 22 | План пользователя |
| 1. Средняя трудоемкость работ в расчете на 1 задачу | Тс1, Тс2 | человеко-часов на 1 задачу | - | 2 | Рассчитывается по данным пользователя |
| 1. Количество часов работы в день | Тч | ч. | 8 | 8 | принято для расчета |
| 1. Ставка налога на прибыль | Нп | % | 24 | 24 |  |

Капитальные затраты на приобретение и использование ПО. Особое значение имеет оценка капитальных затрат на приобретение и использование ПО.

Общие капитальные вложения (Ко) заказчика (потребителя), связанные с приобретением, внедрением и использованием ПО, рассчитываются по формуле:

Ко = Кпр + Кос + Кс + Ктс + Коб, (7.27)

где: Кпр – затраты пользователя на приобретение ПО по отпускной цене разработчика с учетом стоимости услуг по эксплуатации (руб.);

Кос – затраты пользователя на освоение ПО (руб.);

Кс – затраты пользователя на оплату услуг по сопровождению ПО (руб.);

Ктс – затраты на доукомплектацию ВТ техническими средствами в связи с внедрением нового ПО (руб.);

Коб – затраты на пополнение оборотных средств в связи с использованием нового ПО (руб.).

Ко = 5 955 252 + 407 952 + 815 903 = 7 179 107 (руб.)

Расчет экономии основных видов ресурсов в связи с использованием нового ПО

При расчете затрат на заработную плату важно правильно выбрать показатели сравниваемых проектов в зависимости от особенности сервиса. В качестве показателей для сравнения базового и проектируемого ПО можно использовать:

* трудоемкость решаемых задач (на основе хронометражных наблюдений и накопленной статистической информации);
* затраты труда на 100 строк исходного кода;
* количество выполнения транзакций в минуту.

Экономия затрат на заработную плату (Сз) при использовании нового ПО в расчете на объем выполненных работ (см. табл. 7.3 «Исходные данные»):

Сз=Сзе\*А2, (7.28)

где: Сзе – экономия затрат на заработную плату при решении задач с использованием нового ПО в расчете на 1 задачу (руб.);

А2 – объем выполненных работ с использованием нового ПО (задач).

Экономия затрат на заработную плату в расчете на 1 задачу (Сзе):

(7.29)



где: Зсм – среднемесячная заработная плата одного программиста (руб.);

Тс1, Тс2 – снижение трудоемкости работ в расчете на 1 задачу (человеко-часов);

Тч – количество часов работы в день (ч);

Др – среднемесячное количество рабочих дней.

= ((400 000 \* 2)/8)/21 = 4 762 (руб.)



Объем выполняемых работ с использованием нового ПО (100 команд):

А2 = А0 \* Зт2, (7.30)

А2 = 22 \* 100 = 2 200 (100 команд)

Экономия заработной платы при использовании нового ПО (руб.):

Сз=Сзе\*А2, (7.31)

где: Сзе – экономия заработной платы (руб.);

А2 – объем выполняемых работ с использованием нового ПО (100 команд).

Сз = 4 762 \* 2 200 = 10 476 400 (руб.)

Экономия с учетом начисления на зарплату (Сн):

Сн=Сз\* Кнз, (7.32)

где: Сз – экономия заработной платы при использовании нового ПО (руб.);

Кнз – коэффициент начислений на зарплату.

Сн=10 476 400 \* 1,5 = 15 714 600 (руб.)

Общая годовая экономия текущих затрат, связанных с использованием нового ПО (Со) равна экономии с учетом начисления на зарплату (Сн):

Со= 15 714 600 (руб.)

Расчет экономического эффекта. Внедрение нового ПО позволит пользователю сэкономить на текущих затратах, т.е. практически получить на эту сумму дополнительную прибыль. Для пользователя в качестве экономического эффекта выступает лишь чистая прибыль – дополнительная прибыль, остающаяся в его распоряжении (), которая определяется по формуле:



(7.33)



где Нп – ставка налога на прибыль (24%).

= 15 714 600 – ((15 714 600 \* 24%) / 100%) = 11 943 096 (руб.)



В процессе использования нового ПО чистая прибыль в конечном итоге возмещает капитальные затраты. Однако полученные при этом суммы результатов (прибыли) и затрат (капитальных вложений) по годам приводят к единому времени – расчетному году (за расчетный год принят 2006 год) путем умножения результатов и затрат за каждый год на коэффициент приведения который рассчитывается по формуле:



(7.34)



где: Ен – норматив приведения разновременных затрат и результатов;

tр – расчетный год, tр=1;

t – номер года, результаты и затраты которого приводятся к расчетному году.

Норматив приведения разновременных затрат и результатов (Eн) для программных средств ВТ в существующей практике принимается равным 0,15, так как Eн по своей сущности очень близок к ставке рефинансирования НБ РБ. Следовательно, при решении данной задачи коэффициентам приведения (ALFAt) по годам будут соответствовать следующие значения:

* расчетный год

ALFA1 = (1+0,15)1-1=1;

* 2007-й год

ALFA2 =(1+0,15)1-2=0,87;

* 2008-й год

ALFA3 =(1+0,15)1-3=0,66;

* 2009-й год

ALFA4 =(1+0,15)1-4=0,57;

Сведем данные расчета экономического эффекта в таблицу 7.4.

Таблица 7.4. Расчет экономического эффекта от использования нового ПС.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Показатели | Ед. изм. | 2006 | 2007 | 2008 | 2009 |
| Результаты: |  |  |  |  |  |
| - прирост прибыли за счет экономии затрат (Пч) | руб. | 11 943 096 | 11 943 096 | 11 943 096 | 11 943 096 |
| -то же с учетом фактора времени | руб. | 11 943 096 | 10 390 494 | 7 882 443 | 6 807 565 |
| Затраты: |  |  |  |  |  |
| - приобретение, адаптация (Кпр) | руб. | 5 955 252 | - | - | - |
| - освоение ПС (Кос) | руб. | 407 952 | - | - | - |
| - сопровождение ПО (Кс) | руб. | 815 903 | - | - | - |
| Всего затрат: | руб. | 7 179 107 | - | - | - |
| - то же с учетом фактора времени: | руб. | 7 179 107 | 6 245 823 | 4 738 211 | 4 092 091 |
| Экономический эффект: |  |  |  |  |  |
| - превышение рез-та над затратами | руб. | 4 763 989 | 4 144 671 | 3144232 | 2 715 474 |
| - то же нарастающим итогом | руб. | 4 763 989 | 5 383 307 | 6 383 746 | 6 812 504 |
| Коэффициент приведения | ед. | 1,0 | 0,87 | 0,66 | 0,57 |

## 

## Выводы

Затраты на разработку программы у разработчика составили 7 179 107 руб. Разработанное ПО ВТ позволяет получить значительную экономию трудовых, материальных и финансовых расходов в процессе эксплуатации пользователем по сравнению с базовым вариантом программного модуля обработки и отображения картографической информации на цифровой карте местности. Экономия трудовых ресурсов связана с сокращением трудоемкости выполняемых работ, материальных ресурсов – со снижением расходов на материалы (бумагу, магнитные диски), финансовых – с уменьшением денежных расходов (статьи «прочие затраты», «накладные расходы»). Реализация проекта ПО позволит заказчику снизить трудоемкость решения задач и сократить простой сервиса. Все затраты заказчика окупятся в первом году эксплуатации ПО. Проект представляется эффективным и полезным для заказчика.

# Литература

1. Абломейко С.В. Географические информационные системы. Создание цифровых карт: справочное пособие / С.В. Абломейко, Г.П. Апарин, А.Н. Крючков. – Мн.: ИТК НАН РБ, 2000. – 440 с.
2. Шилдт Г. С++. Базовый курс: справочное пособие / Г. Шилдт. – СПб.: Вильямс, 2005. – 788 с.
3. Лаптик Д.В. Библиотека картографического обеспечения: руководство программиста / Д.В. Лаптик. – Мн.: НИИСА, 2003. – 25 с.
4. Палицын В.А. Технико-экономическое обоснование дипломных проектов: методическое пособие в 4-х частях. Часть 4. Проекты программного обеспечения / В.А. Палицын. – Мн.: БГУИР, 2006. – 76 с.
5. Асаёнок И.С. Охрана труда и экологическая безопасность: Методические указания по дипломному проектированию для студентов всех специальностей / И.С. Асаёнок, Т.Ф. Михнюк, Г.М. Дунаева. – Мн.: БГУИР, 1998. – 40 с.
6. Денисенко Г.Ф. Охрана труда: учебное пособие для инж.–экон. спец. вузов. – М.: Высш. шк., 1985. – 319 с.
7. Дядечкин В.П. Психофизиологические резервы повышения работоспособности. – Мн.: Высш. шк., 1990. – 119 с.
8. Среда обитания человека, здоровье, работоспособность: методы оценки и анализа: учебное пособие / И.С. Асаёнок [и др.] – Мн.: БГУИР, 1997. – 86 с.
9. Семич В.П. Охрана труда при работе на персональных электронно-вычислительных машинах и другой офисной технике / В.П. Семич, А.В. Семич. – Мн.: ЦОТЖ, 2001. – 75 с.
10. Голуб В.С. Сборник официальных документов по медицине труда и производственной санитарии. Часть VIII. / В.С. Голуб., С.М. Соколов – Мн.: РЦГЭ МЗ РБ, 2001. – 210 с.
11. Межотраслевая типовая инструкция по охране труда при работе с персональными компьютерами. /Национальный реестр правовых актов Республики Беларусь 14 декабря 2004 г. №8/11835
12. СанПиН 2.2.2. 542 РБ 96. Гигиенические требования к видеодисплейным терминалам, персональным электронно-вычислительным машинам и организации работы.