Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Дисциплина: Программирование на языках высокого уровня

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА к курсовому проекту на тему

АДРЕСНАЯ КНИГА С ПОДДЕРЖКОЙ ФОТОГРАФИЙ  
БГУИР КП 1-40 02 01 012 ПЗ

Студент: группы 150504, Лужков И.А.

Руководитель: Марзалюк А.В.

Минск 2022

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

УТВЕРЖДАЮ

Заведующий кафедрой ЭВМ

Б.В.Никульшин (подпись)

––––––––––––––– 14 сентября 2022г.

ЗАДАНИЕ

по курсовому проектированию

Студенту *Лужкову Ивану Андреевичу ––––––*

1. Тема проекта Адресная книга с поддержкой фотографий

2. Срок сдачи студентом законченного проекта–––––*14 сентября 2022 г.–––*

3. Исходные данные к проекту *Написать программу, которая предоставляет возможность ведения адресной книги с поддержкой фотографий. Изучить методы сохранения данных, самостоятельно выбрать метод для хранения информации о контактах и фотографий.*

4. Содержание расчетно-пояснительной записки (перечень вопросов, которые подлежат разработке) *Титульный лист, Задание по курсовой работе*

*(заполненное и подписанное студентом и преподавателем), Содержание,*

*Введение;*

*1. Обзор методов и алгоритмов решения поставленной задачи;*

*2. Обоснование выбранных методов и алгоритмов;*

*3. Описание программы для программиста;*

*4. Описание алгоритмов решения задачи;*

*5. Руководство пользователя;*

*Заключение;*

*Список использованных источников;*

*Приложение (обязательное): А. листинг кода с комментариями.*

*Б. Скриншоты работы программы.*

5. Перечень графического материала (с точным обозначением обязательных

чертежей и графиков) *1. Диаграмма последовательности.*

6. Консультант по проекту *Марзалюк А.В.*

7. Дата выдачи задания *14 сентября 2022 г.* *.*

8. Календарный график работы над проектом на весь период проектирования (с обозначением сроков выполнения и трудоемкости отдельных этапов):

*Введение, Разделы 1, 2 к 15 октября 2022 г. – 15 %;*

*раздел 3 к 5 ноября 2022 г. – 30 %;*

*разделы 4, 5 к 26 ноября 2022 г. – 30 %;*

*оформление пояснительной записки до 23 мая 2022 г. – 25 %*

*Защита курсового проекта с 27 декабря 2022 г. по 10 июня 2022 г.*

РУКОВОДИТЕЛЬ *– А.В. Марзалюк*

Задание принял к исполнению *\_\_\_\_\_\_\_\_–– И.А. Лужков*

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ ............................................................................................................ 5](file:///C:\Users\vanish\Downloads\Пример_ПЗ%20(1).docx#_page_17_0)

[1ОБЗОР МЕТОДОВ И АЛГОРИТМОВ РЕШЕНИЯ ПОСТАВЛЕННОЙ](file:///C:\Users\vanish\Downloads\Пример_ПЗ%20(1).docx#_page_27_0)

[ЗАДАЧИ.................................................................................................................. 7](file:///C:\Users\vanish\Downloads\Пример_ПЗ%20(1).docx#_page_27_0)

1.1 JSON............................................................................................................. 7

[1.2 XML............................................................................................................. 8](file:///C:\Users\vanish\Downloads\Пример_ПЗ%20(1).docx#_page_31_0)

[2ОБОСНОВАНИЕ ВЫБРАННЫХ МЕТОДОВ ИАЛГОРИТМОВ ............... 12](file:///C:\Users\vanish\Downloads\Пример_ПЗ%20(1).docx#_page_36_0)

[3ОПИСАНИЕПРОГРАММЫДЛЯПРОГРАММИСТА................................. 14](file:///C:\Users\vanish\Downloads\Пример_ПЗ%20(1).docx#_page_39_0)

[4ОПИСАНИЕАЛГОРИТМОВРЕШЕНИЯЗАДАЧИ..................................... 15](file:///C:\Users\vanish\Downloads\Пример_ПЗ%20(1).docx#_page_41_0)

[5РУКОВОДСТВОПОЛЬЗОВАТЕЛЯ................................................................ 19](file:///C:\Users\vanish\Downloads\Пример_ПЗ%20(1).docx#_page_46_0)

[ЗАКЛЮЧЕНИЕ.................................................................................................... 21](file:///C:\Users\vanish\Downloads\Пример_ПЗ%20(1).docx#_page_49_0)

[СПИСОКИСПОЛЬЗОВАННЫХИСТОЧНИКОВ........................................... 22](file:///C:\Users\vanish\Downloads\Пример_ПЗ%20(1).docx#_page_51_0)

[ПРИЛОЖЕНИЕ А Листинг работы с комментариями..................................... 23](file:///C:\Users\vanish\Downloads\Пример_ПЗ%20(1).docx#_page_53_0)

[ПРИЛОЖЕНИЕ Б Скриншоты работы программы…...................................... 24](file:///C:\Users\vanish\Downloads\Пример_ПЗ%20(1).docx#_page_53_0)

***ВВЕДЕНИЕ***

Адресная книга или имя и адресная книга (NAB ) - это книга или база данных , используемые для хранения записей, называемых контактов . Каждая контактная запись обычно состоит из нескольких стандартных полей (например: имя, фамилия, название компании, адрес , телефон номер, e -почта адрес, номер факса , номер мобильного телефона ). Большинство таких систем хранят детали в алфавитном порядке имен людей, хотя в бумажных -содержащих адресных книгах записи могут легко выйти из строя по мере того, как владелец вставляет данные о других людях или по мере того, как люди перемещаются. Во многих адресных книгах используются небольшие папки кольцевых папок , которые позволяют добавлять, удалять и перемещать страницы, чтобы освободить место.

Адресные книги также могут отображаться как программное обеспечение , разработанное для этой цели, например, «Адресная книга» , включенное в Apple Inc. Mac OS X . Простые адресные книги были включены в программное обеспечение электронной почты на протяжении многих лет, хотя в 1990-х годах и позже появились более продвинутые версии; а также в мобильных телефонах .

A менеджер личной информации (PIM) объединяет адресную книгу, календарь , список задач и иногда другие функции.

Записи можно импортировать и экспортировать из программного обеспечения, чтобы передавать их между программами или компьютерами. Общие форматы файлов для этих операций:

* JSON
* XML
* с разделителями табуляции (\* .tab, \* .txt)
* LDIF (\* .ldif, \* .ldi)
* через запятую (\*. csv)
* vCard (\* .vcf)

Отдельные записи часто передаются как vCard (\* .vcf), что примерно сопоставимо с физическим бизнесом. карты . А некоторые программные приложения, такие как Lotus Notes и Open Contacts, могут обрабатывать файл vCard, содержащий несколько записей vCard.

Интернет-адресная книга обычно позволяет пользователям создавать свои собственные веб-страницы (или страницы профиля), которые затем индексируются поисковыми системами, такими как Google и Yahoo. Это, в свою очередь, позволяет другим людям находить пользователей с помощью поиска по их имени, а затем связываться с ними через их веб-страницу, содержащую их личную информацию. Возможность найти людей, зарегистрированных в онлайн-адресных книгах, с помощью поисковой системы обычно зависит от распространенности имени и количества результатов по имени. Как правило, пользователи таких систем могут синхронизировать свои контактные данные с другими известными им пользователями, чтобы обеспечить актуальность их контактной информации.

У многих людей есть много разных адресных книг: их учетные записи электронной почты , их мобильный телефон и "списки друзей" на их социальные сети . Сетевая адресная книга позволяет им организовывать и управлять своими адресными книгами через единый интерфейс и делиться своими контактами в разных адресных книгах и социальных сетях.

На основании вышеизложенного определена тема курсового проектирования как «Адресная книга с поддержкой фотографий» с последующей разработкой программного средства «Contacts».

**1. ОБЗОР МЕТОДОВ И АЛГОРИТМОВ РЕШЕНИЯ  
ПОСТАВЛЕННОЙЗАДАЧИ**

В этом разделе речь пойдет о различных методах сохранения данных.

* 1. **JSON**

JSON— текстовый формат обмена данными, основанный на JavaScript. Как и многие другие текстовые форматы, JSON легко читается людьми. Формат JSON был разработан Дугласом Крокфордом.

Несмотря на происхождение от JavaScript, формат считается независимым от языка и может использоваться практически с любым языком программирования. Для многих языков существует готовый код для создания и обработки данных в формате JSON.

JSON-текст представляет собой (в закодированном виде) одну из двух структур:

* Набор пар ключ: значение. В различных языках это реализовано как запись, структура, словарь, хеш-таблица, список с ключом или ассоциативный массив. Ключом может быть только строка (регистрозависимость не регулируется стандартом, это остаётся на усмотрение программного обеспечения. Как правило, регистр учитывается программами — имена с буквами в разных регистрах считаются разными, например[5]), значением — любая форма. Повторяющиеся имена ключей допустимы, но не рекомендуются стандартом; обработка таких ситуаций происходит на усмотрение программного обеспечения, возможные варианты — учитывать только первый такой ключ, учитывать только последний такой ключ, генерировать ошибку.
* Упорядоченный набор значений. Во многих языках это реализовано как массив, вектор, список или последовательность.

Структуры данных, используемые JSON, поддерживаются любым современным языком программирования, что и позволяет применять JSON для обмена данными между различными языками программирования и программными системами.

В качестве значений в JSON могут быть использованы:

* запись — это неупорядоченное множество пар ключ:значение, заключённое в фигурные скобки «{ }». Ключ описывается строкой, между ним и значением стоит символ «:». Пары ключ-значение отделяются друг от друга запятыми.
* массив (одномерный) — это упорядоченное множество значений. Массив заключается в квадратные скобки «[ ]». Значения разделяются запятыми. Массив может быть пустым, то есть не содержать ни одного значения. Значения в пределах одного массива могут иметь разный тип.
* число (целое или вещественное).
* литералы true (логическое значение «истина»), false (логическое значение «ложь») и null.
* строка — это упорядоченное множество из нуля или более символов юникода, заключённое в двойные кавычки. Символы могут быть указаны с использованием escape-последовательностей, начинающихся с обратной косой черты «\» (поддерживаются варианты \", \\, \/, \t, \n, \r, \f и \b), или записаны шестнадцатеричным кодом в кодировке Unicode в виде \uFFFF.

Следующий пример показывает JSON-представление данных об объекте, описывающем человека. В данных присутствуют строковые поля имени и фамилии, информация об адресе и массив, содержащий список телефонов. Как видно из примера, значение может представлять собой вложенную структуру.

{

**"firstName"**: "Иван",

**"lastName"**: "Иванов",

**"address"**: {

**"streetAddress"**: "Московское ш., 101, кв.101",

**"city"**: "Ленинград",

**"postalCode"**: 101101

},

**"phoneNumbers"**: [

"812 123-1234",

"916 123-4567"

]

}

В качестве значений в JSON могут быть использованы как числа, так и строки. Поэтому запись "postalCode": "101101" содержит строку, а "postalCode": 101101 — уже числовое значение. Из-за слабой типизации в JavaScript и PHP строка может быть приведена к числу и не влиять на логику программы. Тем не менее, рекомендуется аккуратно обращаться с типом значения, так как JSON служит для межсистемного обмена.

* 1. **XML**

XML — «расширяемый язык разметки». Спецификация XML описывает XML-документы и частично описывает поведение XML-процессоров (программ, читающих XML-документы и обеспечивающих доступ к их содержимому). XML разрабатывался как язык с простым формальным синтаксисом, удобный для создания и обработки документов как программами, так и человеком, с акцентом на использование в Интернете. Язык называется расширяемым, поскольку он не фиксирует разметку, используемую в документах: разработчик волен создать разметку в соответствии с потребностями к конкретной области, будучи ограниченным лишь синтаксическими правилами языка.

Расширение XML — это конкретная грамматика, созданная на базе XML и представленная словарём тегов и их атрибутов, а также набором правил, определяющих, какие атрибуты и элементы могут входить в состав других элементов. Сочетание простого формального синтаксиса, удобства для человека, расширяемости, а также базирование на кодировках Юникод для представления содержания документов привело к широкому использованию как, собственно, XML, так и множества производных специализированных языков на базе XML в самых разнообразных программных средствах.

C физической точки зрения документ состоит из сущностей (англ. entities), из которых каждая может ссылаться на другую сущность. Единственный корневой элемент — документная сущность. Содержание сущностей — символы.

С логической точки зрения документ состоит из комментариев (англ. comments), объявлений (англ. declarations), элементов (англ. elements), ссылок на сущности (англ. character references) и инструкций обработки (англ. processing instructions). Всё это в документе структуризуется разметкой (англ. markup).

Сущность — мельчайшая часть в документе. Все сущности что-нибудь содержат, и у всех них есть имя (существуют исключения, напр. документная сущность). Проще говоря, термин «сущность» описывает «сущую вещь», «что-то».

Документ состоит из сущностей, содержание которых — символы. Все символы разделены на два типа: символы данных (англ. character data) и символы разметки. К разметке относятся:

* теги (англ. tags) <- обозначают границы элементов
* объявления и инструкции обработки, включая их атрибуты (англ. attributes)
* ссылки на сущности
* комментарии
* а также последовательности символов, обрамляющие секции «CDATA»

Часть документа, не принадлежащая разметке, составляет символьные данные документа.

На языке XML структура из прошлого раздела выглядела бы примерно так:

**<person>**

**<firstName>**Иван**</firstName>**

**<lastName>**Иванов**</lastName>**

**<address>**

**<streetAddress>**Московское ш., 101, кв.101**</streetAddress>**

**<city>**Ленинград**</city>**

**<postalCode>**101101**</postalCode>**

**</address>**

**<phoneNumbers>**

**<phoneNumber>**812 123-1234**</phoneNumber>**

**<phoneNumber>**916 123-4567**</phoneNumber>**

**</phoneNumbers>**

**</person>**

или так:

**<person** firstName="Иван" lastName="Иванов"**>**

**<address** streetAddress="Московское ш., 101, кв.101" city="Ленинград" postalCode="101101" **/>**

**<phoneNumbers>**

**<phoneNumber>**812 123-1234**</phoneNumber>**

**<phoneNumber>**916 123-4567**</phoneNumber>**

**</phoneNumbers>**

**</person>**

**2.ОБОСНОВАНИЕ ВЫБРАННЫХ МЕТОДОВ И АЛГОРИТМОВ**

Для выполнения данной задачи был выбран формат файла JSON.

Основная причина выбора данного формата файла является то, что он удобен как в чтении и создании файлов человеком, так и работой с этим форматом в современных языках программирования.

В данной курсовой работе нам нужен формат файла, который :

* занимает меньше места
* удобен в изменении, создании и работе с файлами на языке программирования с++ на фреймворке Qt
* может хранить числа и строки
* удобочетаем для человека

Так как мы будем хранить информацию о контактах человека, а их может быть большое количество( примерно от 130 до 200),то один из ключевых факторов в выборе формата файла для сохранения в него данных это компактность(файл занимает меньше места чем аналоги).

Второй ключевой фактор – это сложность работы с этим форматом на языке c++ с использованием фреймворка Qt.

В фреймворке Qt есть инструменты как для работы с форматом JSON, так и с XML. Они представляют собой различные классы для чтения, записи, изменения, просмотра и парсинга JSON и XML файлов. В данном фреймворке работа с JSON и XML форматами одинаково проста с точки зрения разработки адресной книги.

Ключевое различие между JSON и XML заключается в том, что JSON имеет меньший размер файла по сравнению с XML. JSON обращается к данным через объекты JSON, тогда как XML требует анализа данных. JSON легко читается, поскольку имеет более организованную структуру кода. С другой стороны, XML трудно интерпретировать из-за его сложной структуры.

Так же формат JSON намного легче читается для человека, чем данные в виде XML формата, что видно на примере одной и той же структуры на JSON :

{

**"firstName"**: "Иван",

**"lastName"**: "Иванов",

**"address"**: {

**"streetAddress"**: "Московское ш., 101, кв.101",

**"city"**: "Ленинград",

**"postalCode"**: 101101

},

**"phoneNumbers"**: [

"812 123-1234",

"916 123-4567"

]

}

И в виде XML :

**<person>**

**<firstName>**Иван**</firstName>**

**<lastName>**Иванов**</lastName>**

**<address>**

**<streetAddress>**Московское ш., 101, кв.101**</streetAddress>**

**<city>**Ленинград**</city>**

**<postalCode>**101101**</postalCode>**

**</address>**

**<phoneNumbers>**

**<phoneNumber>**812 123-1234**</phoneNumber>**

**<phoneNumber>**916 123-4567**</phoneNumber>**

**</phoneNumbers>**

**</person>**

Подводя итог, JSON был выбран по причине хорошей читаемости для человека(что может помочь при проблемах с повреждениями файла и др.) а также занимает меньше места по сравнению с XML.

**3. ОПИСАНИЕ ПРОГРАММЫ ДЛЯ ПРОГРАММИСТА**

Диаграмма последовательности — UML-диаграмма, на которой для некоторого набора объектов на единой временной оси показан жизненный цикл объекта (создание-деятельность-уничтожение некой сущности) и взаимодействие актеров (действующих лиц) информационной системы в рамках прецедента [5].

Основными элементами диаграммы последовательности являются обозначения объектов (прямоугольники с названиями объектов), вертикальные «линиижизни»(англ.lifeline), отображающие течение времени, прямоугольники, отражающие деятельность объекта или исполнение им определенной функции (прямоугольники на пунктирной «линии жизни»), и стрелки, показывающие обмен сигналами или сообщениями между объектами. Диаграмма последовательности разработанного приложения представлена на рисунке 3.1.