Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования "Южный федеральный университет"

Институт компьютерных технологий и информационной безопасности

ОТЧЕТ

о выполнении индивидуального задания

по дисциплине

"Основы алгоритмизации программирования"

Выполнил: ст. гр. КТбоX-X

Иванов И.И.,

Проверил: зав. кафедрой МОП ЭВМ

Хусаинов Н.Ш.

Таганрог, 2016

**1. Постановка задачи**

Требования к алгоритму функционирования приложения, описание форматов входных и выходных данных, критерии оценки приложения приведены в выданном преподавателем задании (в данном документе не дублируются).

**2. Проектирование архитектуры приложения**

*В данном разделе рекомендуется привести блок-схемы, построенные исполнителем в ходе проектирования приложения.*

*Варианты блок-схем:*

*- блок-схема, содержащая укрупненное описание алгоритма функционирования приложения. Элементы этой блок-схемы могут соответствовать, например, впоследствии созданным функциям программы на языке программирования. Допускается именовать блоки, описывающие предопределенные процессы, именами функций, при этом либо в самой блок-схеме, либо после нее должны быть приведены краткие описания каждого такого блока;*

*- блок-схема, содержащая детализированное описание алгоритма функционирования приложения на одном из этапов обработки данных. Как правило, разумным является привести здесь блок-схему, соответствующую одному из предопределенных блоков из блок-схемы укрупненного уровня (т.е. фактически – алгоритму функционирования одной из функций программы). Выбор такого блока следует осуществлять так, чтобы, с одной стороны, не "скатиться" к блок-схеме из 1-2 блоков (кроме Начало и Конец), а с другой – чтобы блок-схема не занимала больше одной страницы. И в одном, и в другом случае это может свидетельствовать о не совсем удачном разбиении программы на подпрограммы.*

*Блок-схемы должны быть оформлены с учетом требований ГОСТ (как мы делали на практических занятиях) и соответствовать парадигме структурного программирования. Приветствуется краткое текстовое описание блок-схемы, но не в плане того, как работает алгоритм (это должно быть понятно из блок-схемы), а, содержащее, например:*

*- описания используемых переменных (кроме, конечно, счетчиков циклов);*

*- описание каких-либо ключевых моментов работы алгоритма (которые, по мнению разработчика, проблематично отразить стандартными блоками блок-схемы);*

*- перечень используемых подпрограмм (в виде предопределенных блоков), в том числе библиотечных.*

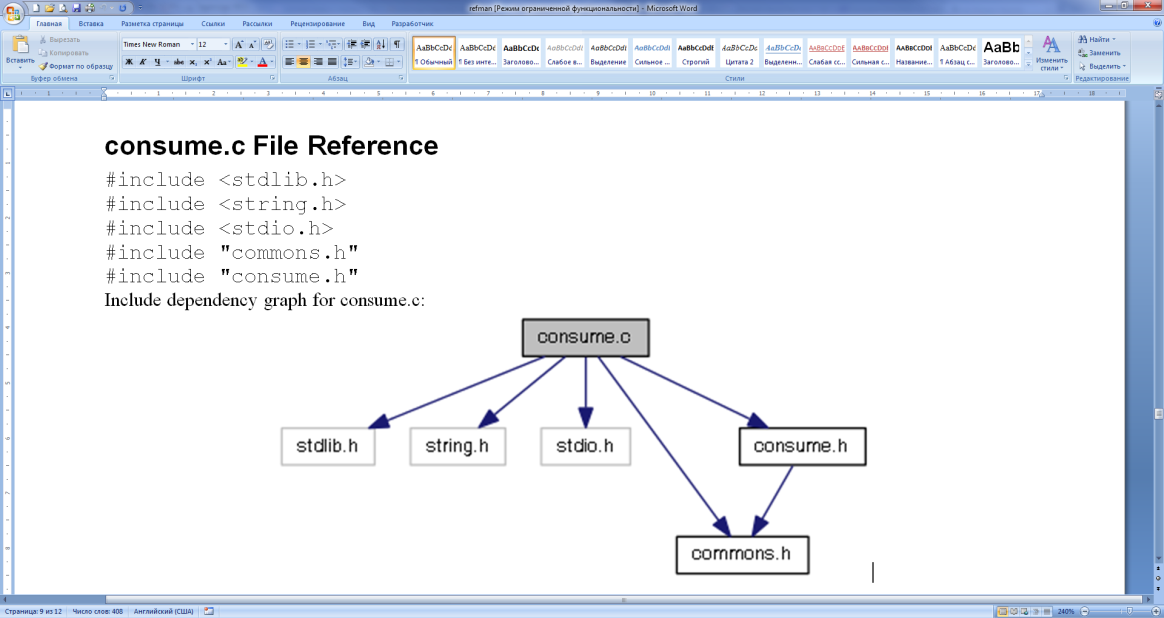
*Для разработки блок-схем могут использоваться любые инструментальные средства, позволяющие интегрировать разработанную блок-схему в итоговый документ формата doc/docx/pdf.*

**3. Модульная архитектура приложения**

*В данном разделе рекомендуется привести перечень модулей приложения. Под модулем, как правило, понимается совокупность h- и c/cpp-файла.*

*Для каждого модуля (пары h- и c/cpp-файла) следует указат его назначение (для работы с какими объектами предназначен – не на уровне типов данных, а на уровне логики приложения), какие операции над этими объектами позволяет определять.*

*Раздел целесообразно завершить графом подключения файлов. Такой граф может иметь представление, как показано на рисунке 3.1.*

**

*Рисунок 3.1 – Граф подключений файлов в проекте Test*

**4. Описание объектов программы**

*В данном разделе следует описать программные объекты: макросы, введенные типы данных, переменные, функции, типы данных).*

*Для объектов-данных и типов желательно указать (в рамках возможного) его структуру, тип данных, область видимости, в каком модуле описан(а).*

*Для функций – охарактеризовать назначение, формальные параметры, возвращаемое значение, при необходимости детализировать внутренний алгоритм (если он требует пояснений) и локальные переменные, указать в каком модуле описана, область видимости.*

*Группировать описания следует либо по модулям (а затем – по типам объектов), либо по типам объектов. Но порядок группировки должен быть единым для объектов всего проекта.*

*Раздел должен завершаться графом вызовов подпрограмм, который может иметь вид, подобный тому, что показан на рисунке ниже. Такой граф может быть сформирован в автоматизированном режиме специальными инструментальными средствами или же нарисован разработчиком (в том числе от руки и затем отсканирован). Граф строится на основе статического анализа кода, т.е. не надо запускать программу, надо только просмотреть какая функция из какой может быть вызвана. Условия вызова тоже указывать не надо (этим граф вызовов отличается от обобщенной блок-схемы). Естественно, что начальной всегда будет функция main() – из нее будут только исходящие стрелки.*



*Рисунок 4.1 – Граф вызовов функций в проекте Test*

**5. Оценка требований к аппаратным средствам для работы приложения**

*В таком разделе обычно приводится оценка требований к ресурсам ЭВМ, на которой должно выполняться приложение. Для нашего проекта достаточно будет оценить:*

*- примерные требования к объему памяти для хранения данных (для случая теста максимальной размерности) – суммарный объем памяти, выделяемый под структуры и массивы либо статически, либо динамически (Кб, Мб) – с обоснованием оценки, т.е. расчетом – под какой массив сколько памяти надо. Память под отдельные переменные обычно не считается или оценивается "на глазок, с запасом";*

*- примерный размер свободного места на диске для установки (копирования) приложения и минимального набора файлов с исходными данными, входными и выходными данными.*

**6. Листинг программы**

*В этом разделе приводится листинг приложения. "Идеально" оформленный листинг приложения отвечает следующим требованиям:*

*- перед началом каждого файла – наименование этого файла;*

*- c/cpp-файл должен следовать за соответствующим ему заголовочным файлом;*

*- подсветка синтаксиса в соответствии с требованиями языка C/C++ (см., например, текстовый редактор Notepad++ - пример сгенерирован с его использованием);*

*- нумерация строк в пределах каждого файла;*

*- для отображения символов желательно выбирать не пропорциональный, какой-либо моноширинный шрифт – это шрифт, все кегельные площадки знаков которого имеют одинаковую ширину (например –* Courier New*).*

*Следовать всем указанным требованиям вовсе необязательно, но привести распечатку всех файлов, разработанных исполнителем, необходимо.*

**Файл new2.c**

#include <stdio.h>

int main**(**void**)**

**{**

int i **=** 3**,** j **=** i **-** 2**;**

**switch** **(**i **+** 2**)** **{**

**case** 1**:** j**++;**

**case** 2**:** j**++;**

**default:**j **=** 0**;**

**case** 0**:** j**++;** **break;**

**}**

printf**(**"%d"**,** j**);**

**return** 0**;**

**}**

*Внимание!*

*1) Ко всем приведенным выше требованиям следует относится как к рекомендациям.*

*2) Если какая-то часть информации, которая требуется в отчете, в явном виде указана в листинге программы (например, в комментариях), то ничего страшного, если в отчете она будет отсутствовать (в наибольшей степени это может относиться к содержанию разделов 3-4). Однако следует придерживаться единого подхода к представлению информации: будет хуже всего, если по одним модулям (или функциям) информация будет представлена в описательных разделах отчета, а по другим модулям (функциям) – сконцентрирована в комментариях в листингах. В общем, пусть лучше меньше, но чтобы в рамках проекта было единообразие представления сведений о приложении.*

*3) Обратите внимание, что рисовать блок-схемы, графы и т.п. совсем необязательно в каких-либо редакторах – если вам так удобнее и быстрее, то это можно сделать от руки, сфотографировать в нормальном качестве и вставить в отчет.*

*4) Итоговый формат файла с отчетом – только pdf. Файлы другого типа открываться не будут.*

*5) При необходимости в отчет может быть включена дополнительная информация по желанию исполнителя.*