ПЛАН ПРОЕКТА

1. Описание процесса планирования

Процесс разработки состоит из пяти пунктов:

Таблица 1 - Пункты проекта

Пункты	Календарные сроки
«Белая книга»	09.02 – 27.02
План проекта	28.03 – 14.03
Бета-версия	15.03 – 5.04
RC-версия	6.04 – 2.05
Финальный релиз	3.05 – 23.05

Раз в неделю, по средам проводится обсуждение качества проведения спринта, заслушиваются отчеты о проделанной работе.

2. Описание процесса специфицирования

Основные особенности языка «Крабик (V)_ii_(V)» описаны в «Белой книге».

3. Описание процесса разработки

Наиболее удобной на наш взгляд является методология Scrum. Удобна она по следующим причинам:

- Каждый промежуток между СР будет спринтом, за время которого мы реализуем задание, поставленное в СР,
- Возможности ПО к реализации определены в начале каждого СР,
- Подход гибок и полностью покрывает наши необходимости.

Список спринтов представлен в таблице 2.

Таблица 2 - Список спринтов

Название Спринта	Календарные сроки
«Белая книга»	09.02 – 27.02
План проекта	28.02 – 14.03
Бета-версия	15.03 – 5.04
RC-версия	6.04 – 2.05
Финальный релиз	3.05 – 23.05

Все спринты длятся от 2х до 4х недель, согласно методологии Scrum. Что делает процесс разработки достаточно гибким и быстрым.

4. Описание процесса тестирования

Схема тестирования будет следующей: тестироваться будет каждый результат спринта в отдельности, а в конце проекта будет финальное тестирования всех вместе взятых результатов спринтов. В свою очередь каждый Спринт будет делиться на модули по усмотрению тестировщика для облегчения процесса тестирования, после поочередной тестировки всех модулей, производится финальное тестирование спринта.

5. Стандарт кодирования

Используется стандарт "NASA C style guide" (доступен по ссылке http://homepages.inf.ed.ac.uk/dts/pm/Papers/nasa-c-style.pdf), некоторые основные положения написаны ниже.

5.1. Отступы, пробелы и пустые строки

В файле следует выделять "параграфы" несколькими пустыми строками.

```
#define LOWER 0
#define UPPER 300
#define STEP 20


main() /* Fahrenheit-Celsius table */
{
        int fahr;
        for (fahr = LOWER; fahr <= UPPER; fahr = fahr + STEP)
                printf(“%4d %6.1f\n”, fahr, (5.0/9.0)*(fahr - 32));
}
```

В выражениях следует отделять операторы и операнды друг от друга пробелом.

```
*average = *total / *count; /* compute the average */
```

Для отступов следует использовать табуляцию.

```
main()
{
        int c;
        c = getchar();
        while (c!= EOF)
```

```
    {
        putchar(c);
        c = getchar();
    }
}
```

5.2. Комментарии

При написании кода следует не забывать про комментарии. Обозначим несколько необходимых "уровней" комментирования:

1) На уровне проекта. В readme файле проекта необходимо обозначить общее описание проекта.

2) На уровне файла необходимо написать пролог, в котором обозначено назначение данного файла и другая необходимая информация.

3) На уровне функции пролог для функции, в котором описана функция и ее действия.

4) По всему файлу строчные комментарии, помогающие понять назначение переменных и смысл некоторых конструкций.

Виды комментариев:

1) Пролог файла

```
/*****************************************************
 * FILE NAME                                         *
 *                                                   *
 * PURPOSE                                           *
 *                                                   *
 *****************************************************/
```

2) Комментарий блока

```
/*
 * Write the comment text here, in complete sentences.
 * Use block comments when there is more than one
 * sentence.
 */
```

3)Комментарии в коде

```
double ieee_r[];      /* array of IEEE real*8 values */
unsigned char ibm_r[];    /* string of IBM real*8 values */
int count;                /* number of real*8 values */
```

5.3. Наименования

Имена файлов, функций, констант и переменных должны быть хорошо читаемыми.

Следует придерживаться следующих правил при выборе имени:

1) Имя должно отражать то, как используется элемент в программе.

2) В случае, если есть несколько функций связанных с одной сущностью, следует использовать сокращения. К примеру, если функция относится к таблицы символов следует использовать префикс "st_".

3) Недопустимо использование одинаковых имен для переменных и структур.

4) Для наименования переменных следует использовать символы нижнего регистра, разделенные нижним подчеркиванием.

5) При наименований функций каждое слово должно начинаться с большой буквы. Использовать нижнее подчеркивания не стоит.

6) Для констант следуют использовать символы верхнего регистра, разделение слов с помощью нижнего подчеркивания.

open_database variables

ProcessError function names

MAX_COUNT constants

5.4. Организация файлов

Любой файл должен начинаться с пролога. В прологе следует написать имя файла и его назначение.

```
/*****************************************************************
**********
* FILE NAME: qwe.c                                              *
* PURPOSE: this file was created just for fun                   *
*****************************************************************
***********/
```

5.5. Суффиксы файлов

Таблица 3 - Суффиксы

Тип файла	Суффикс
Исходный файл C	.c
Объектный файл	.o
Заголовочный файл	.h
Файл YACC	.y
Файл LEX	.l
Makefile	.mak

5.6. Организация функций

Перед каждой функций следует написать пролог, в котором будет описано предназначение функции, ее аргументы и возвращаемое значение.

Пример определения функции:

```
int getline (char *str, int length)
{
    ...
}
```

В начале функции следует определить все переменные, используемые в функции.

5.7. Переменные

Переменные одного типа стоит объявлять на разных строках. Напротив определения следует написать краткий комментарий.

```
int x; /* comment */
int y; /* comment */
```

5.8. Структуры

Объявлять структуры следует в следующем стиле:

```
typedef struct symbol
{
    char *name;
    int type;
    int flags;
    int value;
} symbol_type;
symbol_type symbol_table[NSYMB];
```

5.9. Формат операций

Не стоит выделять пробелами следующие операторы:

```
p->m s.m a[i]
```

При вызове функции не надо ставить пробелы около скобок.

```
exp(2, x)
```

Не надо ставить пробелы между унарными операторами и их операндами.

```
!p ~b ++i -n *p &x
```

Стоит всегда вставлять пробел после оператора приведения типа.

```
(long) m
```

Всегда выделяйте пробелами оператор присваивания.

```
c1 = c2
```

Всегда вставляйте пробел в условных операторах.

```
z = (a > b) ? a : b;
```

После запятых и точек с запятой должен быть пробел или символ переноса строки.

```
strncat(t, s, n)
for (i = 0; i < n; ++i)
```

Остальные операторы должны выделяться пробелами.

```
x + y a < b && b < c
```

5.10. Выражения

В каждой строке должно быть только одно выражение.

```
switch (axescolor)
{
    case 'B':
        color = BLACK;
        break;
```

```
        case 'Y':
                color = YELLOW;
                break;
        case 'R':
                color = RED;
                break;
        default:
                color = GREEN;
                break;
}
```

Открывающая фигурная скобка всегда переносится на следующую строку.

Для блочных выражений if, for и т.д. рекомендуется всегда использовать фигурные скобки.

```
for (i = 0, j = strlen(s)-1; i < j; i++, j--)
{
        c = s[i];
        s[i] = s[j];
        s[j] = c;
}
```

6. Календарь проекта

В таблице 4 представлен календарь проекта.

Таблица 4 - Календарь проекта

Название	Длительность	Начало	Окончание	Участники
Введение в проект	1	08.02.2015	09.02.2015	Прозоров, Цирюльников, Шатров
"Белая книга"	15	09.02.2015	27.02.2015	Прозоров, Цирюльников, Шатров
Планирование	12	28.02.2015	14.03.2015	Прозоров, Цирюльников, Шатров
Бета Версия	17	15.03.2015	05.04.2015	Прозоров, Цирюльников
Релиз-Кандидат	20	06.04.2015	02.05.2015	Прозоров, Цирюльников
Финальный Релиз	17	03.05.2015	23.05.2015	Прозоров, Цирюльников
Тестирование	52	15.03.2015	23.05.2015	Шатров
Документация	52	15.03.2015	23.05.2015	Шатров
Защита	2	20.05.2015	21.05.2015	Прозоров, Цирюльников, Шатров

7. Роли и ответственности участников проекта

Прозоров Никита - ведущий разработчик проекта.

Цирюльников Алексей - тимлид, разработчик.

Шатров Степан - тестировщик, писарь.

8. Описание технологического окружения

- Язык реализации – C,
- Стандарт языка – C99,
- Версия компилятора – 4.8.2,
- Генератор синтаксических анализаторов – GNU Bison, версия 3.0.2,
- Генератор лексических анализаторов – Flex, версия 2.5.35,
- Система сборки – GNU make, версия - 3.81,
- Система контроля версий – GitHub,
- Система бак-трекинга – GitHub.

9. Журнал проекта

Таблица 5 - журнал проекта

Задача	Календарные сроки	Ответственный участник команды
«Белая книга»	9 февраля – 27 февраля	Прозоров, Цирюльников, Шатров
Планирование проекта	28 февраля – 14 марта	Прозоров, Цирюльников, Шатров