ПЕТРОЗАВОДСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
КАФЕДРА ИНФОРМАТИКИ И МАТЕМАТИЧЕСКОГО ОБЕСПЕЧЕНИЯ

Отчёт по дисциплине
«Верификация ПО»

Выполнил:
студент группы 22609 А. А. Шаповалов

Лектор:
к.т.н., доцент К. А. Кулаков

Петрозаводск
2014

1. Выбор и согласование тематики проекта

Проект представляет собой программу, выполняющую оптимизацию работы склада целлюлозно-бумажного комбината. После того, как рулон был произведён, он некоторое время хранится на складе, а потом отправляется покупателю. Рулоны по складу перемещаются с помощью автомобилей-погрузчиков. Склад состоит из ячеек. В каждой ячейке могут храниться рулоны. Рулоны складываются в несколько штабелей. Рулоны в каждом штабеле и штабели в ячейке действуют по принципу LIFO (последним зашёл - первым вышел). Один рулон можно размещать сверху другого только если диаметр верхнего не превышает диаметр нижнего. У каждой ячейки есть максимальная высота, которую могут занимать рулоны (более формально - ограничение на высоту верхней части самого верхнего рулона). Также есть ограничение на высоту нижней части самого верхнего рулона - оно связано с максимальной высотой, на которую можно поднять рулон при помощи погрузчика.

В ходе выполнения данной работы рассматривалась программа, выполняющей оптимизацию работы склада для хранения рулонов целлюлозно-бумажного комбината. Основная задача, которую необходимо решать - размещение рулона. На склад поступает рулон, и нужно сообщить, в какую ячейку его следует разместить.

2. Описание функциональности.

Заявляется и будет протестирована следующая функционаьлность:

1. Прочитать из входного файла информацию о складе, о плане выработки и о свежеприбывшем рулоне
2. Определить ячейку, в которую следует разместить рулон
    - Одинаковые рулоны будут храниться вместе в одних ячейках
    - Если заполнено менее 94% склада, можно выделять пустую ячейку для нового рулона. Если нет - пустую ячейку нужно использовать только если можно сразу занять её целиком.

- Если рулон не удаётся поместить к однотипным, нужно попытаться поместить его в сборную ячейку для разнотипных рулонов

3. Вывести ответ

3. Описание вариантов использования программного обеспечения.

Программа предназначена для решения задач оптимизации склада целлюлозно-бумажного предприятия. Пользователь может составлять входные данные вручную (например, при тестировании) или автоматически (при поступлении информации с системы управления складом).

4. Разработка проекта архитектуры.

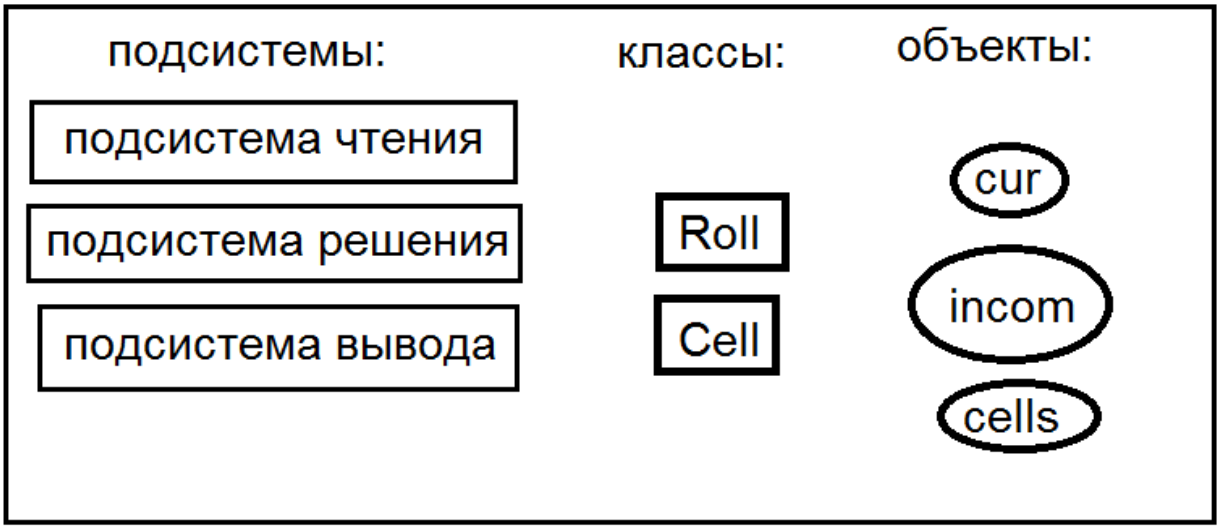

Рис. 1: Архитектура

Пользователь - это клиент, который будет передавать информацию о складе во входной файл и читать ответ из выходного файла. Нет необходимости тестировать это взаимодействие

Подсистема чтения осуществляет чтение входного файла. Будет протестировано.

Подсистема решения осуществляет решение поставленной задачи. Будет протестировано.

Подсистема вывода осуществляет вывод результата в выходной файл. Будет протестировано.

Объекты классов Cell и Roll используются для хранения информации о ячейках и рулонах. В объекте класса Roll записаны свойства рулона, в объекте класса Cell - свойства ячейки (в том числе содержащиеся в ячейке рулоны). Сам склад представлен как массив объектов класса Cell. Эти классы будут протестированы как по отдельности (модульное тестирование), так и в связке (интеграционное).

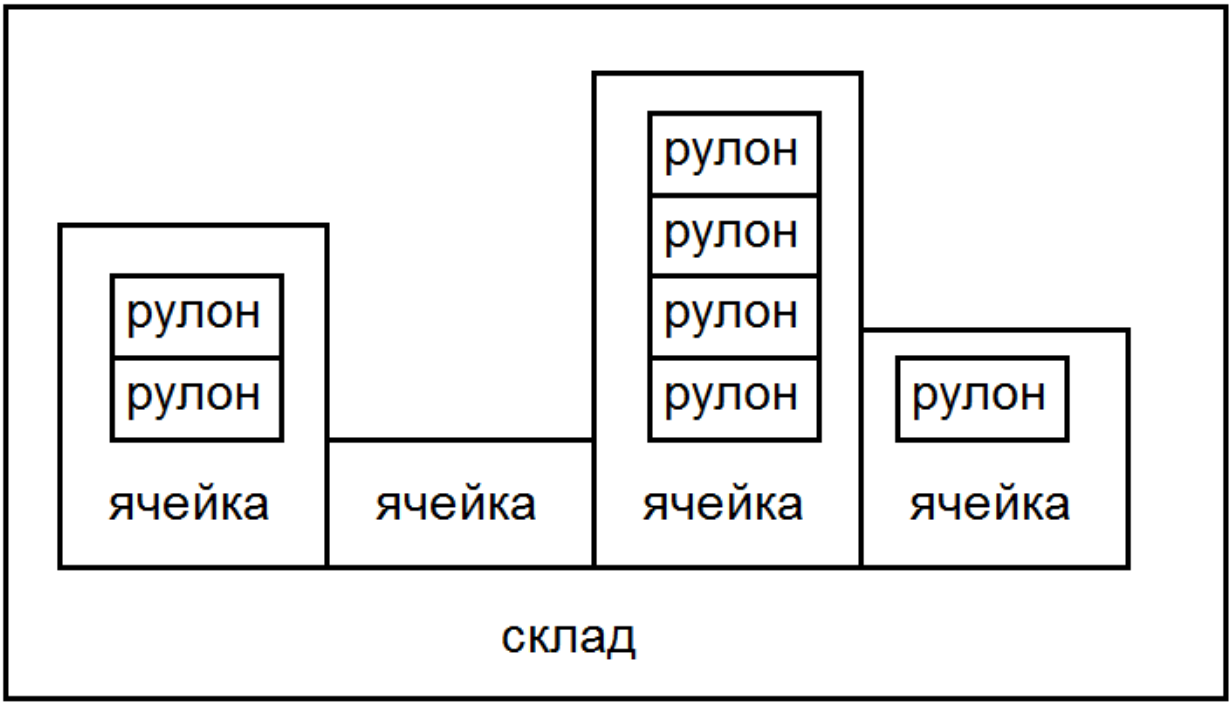

Рис. 2: Представление склада в программе

4.1 Класс Roll.

Объект этого класса хранит информацию о рулоне.

Поля:

```
long long     material;     //    материал
int           market_type;  //    Вид рынка (Экспорт)
string        konosament;   //    Коносамент (коносамент)
string        sort;         //    Сорт (Сорт)
string        mark;         //    Мрака (Марка)
string        paper_type;   //    Вид картона/бумаги (Вид)
int           height;       //    Высота рулона (Формат)
int           diameter;     //    Диаметр рулона  (нормативная диаметр)
int           density;      //    Плотность (Масса м2)
```

Методы, которые будут тестироваться:

```
bool read( istream &);
void print(ostream &out) const;
```

4.2 Класс Cell.

В этом классе хранится максимальная высота, на которую погрузчик может поднять рулон:

```
static const int max_avail_height = 2800;      // максимальная доступная высота
```

Объект этого класса хранит информацию о ячейке.

Поля:

```
vector< vector<int> > pile;       // эмулирование штабелей
vector<Roll> rolls;               // список рулонов в ячейке
int max_stack_cnt;                // количество штабелей
int max_stack_height;             // максимальная высота штабеля
```

Методы, которые будут тестироваться:

```
/* можем ли мы поставить в текущую ячейку рулон cur */
bool can_push(const Roll&, const bool force_new_stack) const;

/* Совпадает ли по типу рулон cur с верхним рулоном в ячейке */
bool is_like_last(const Roll &cur) const;

/* Совпадает ли по типу рулон cur с остальными рулонами в ячейке */
bool is_like_all(const Roll &cur) const;

/* Возвращает количество рулонов в ячейке */
int rolls_cnt() const;

/* прочитать из входного потока in характеристики ячейки */
bool read(istream&);

/* вывести в выходной поток характеристики ячейки */
void print(ostream&) const;
```

4.3 Программа.

Данная система делится на подсистемы чтения, решения и вывода. Содержит следующие объекты:

```
vector<Cell> cells;               // Описание незаблокированных ячеек
Roll cur;                         // Свежеприбыший рулон
vector <pair<Roll, int> > incom;  // Список рулонов, которые привезут на склад
```

4.4 Подсистема чтения.

Данная подсистема предназначена для считывания информации из входного файла.

Методы, которые будут тестироваться:

```
bool read_put_roll(vector<Cell> &cells, Roll &cur)
```

4.5 Подсистема решения.

Данная подсистема предназначена для решения задачи размещения рулона. Подсистема будет тестироваться вручную с использованием отладчика Visual Studio. Будут при необходимости исправлены или заменены входные данные. При помощи того же отладчика Visual Studio будет проверятся ответ.

4.6 Подсистема вывода.

Данная подсистема предназначена для вывода ответа. Подсистема будет тестироваться.

5. Разработка модели потоков данных.

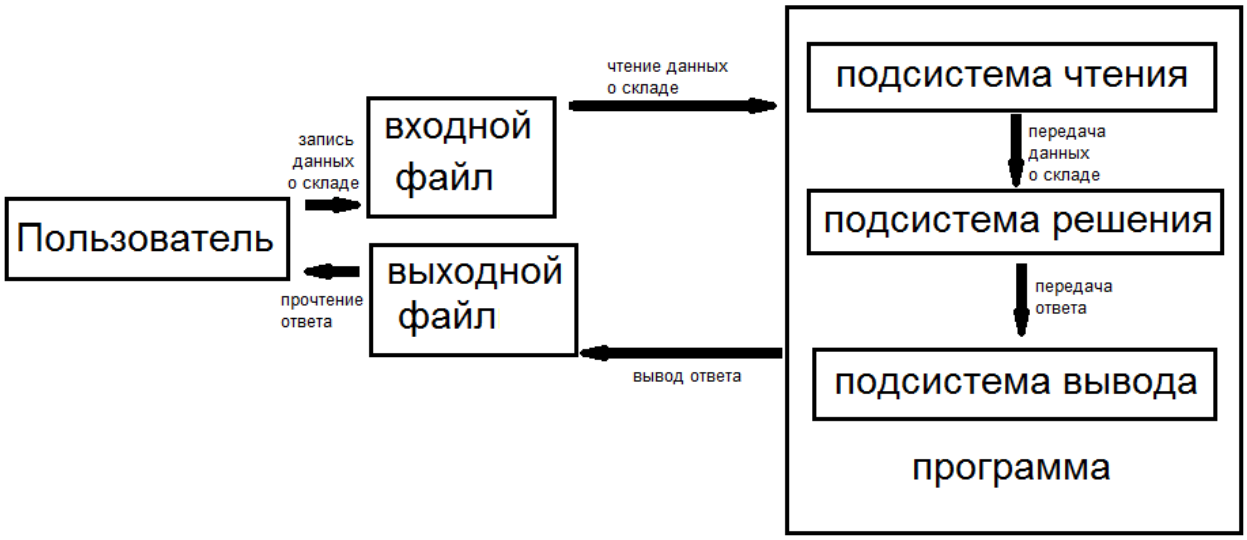

Рис. 3: схема потоков данных

6. План тестирования

Модульное тестирование применяется к классам Roll и Cell. При модульном тестировании класса Cell вместо объектов класса Roll будут использоваться заглушки - числа типа int. При интеграционном тестировании класса Cell с классом Roll будут использоваться те же тесты, что и при модульном тестировании класса Cell, но при модульном номера будут с суффиксом m, а при интеграционном - с суффиксом i. Затем будет произведено модульное тестирование подсистем. После него - системное тестирование. Модульное тестирование системы решения и системное тестирование будут проводиться на одних и тех же тестах, для модульного тестирования системы решения будет использоваться суффикс m, для системного тестирования суффикс s.

6.1 Схема интеграции:

Интеграция будет проводится в следующем порядке:

1. Roll + Cell
2. (Roll + Cell)+ подсистема решения
3. Подсистема ввода + ((Roll + Cell)+ подсистема решения) + подсистема вывода

6.2 Подход к тестированию

Тестирование будет проводиться в автоматическом режиме с помощью самописной утилиты, которая запускает программу на всех тестах. Утилита, исполняемый файл программы т тесты должны располагаться в одной папке. Файлы с тестами должны быть названы inputX.txt, где X - номер теста. Тестовые файлы пронумерованы последовательными натуральными числами от единицы, ведущих нулей быть не должно. Каждому файлу-тесту соответствует файл с ответом answerX.txt. Для запуска модульных тестов добавлен специальный модуль тестирования в программу. Тестировщику необходимо запустить утилиту-тестировщик и ожидать результат. В случае ошибки тестировщику нужно будет составить отчёт об ошибке.

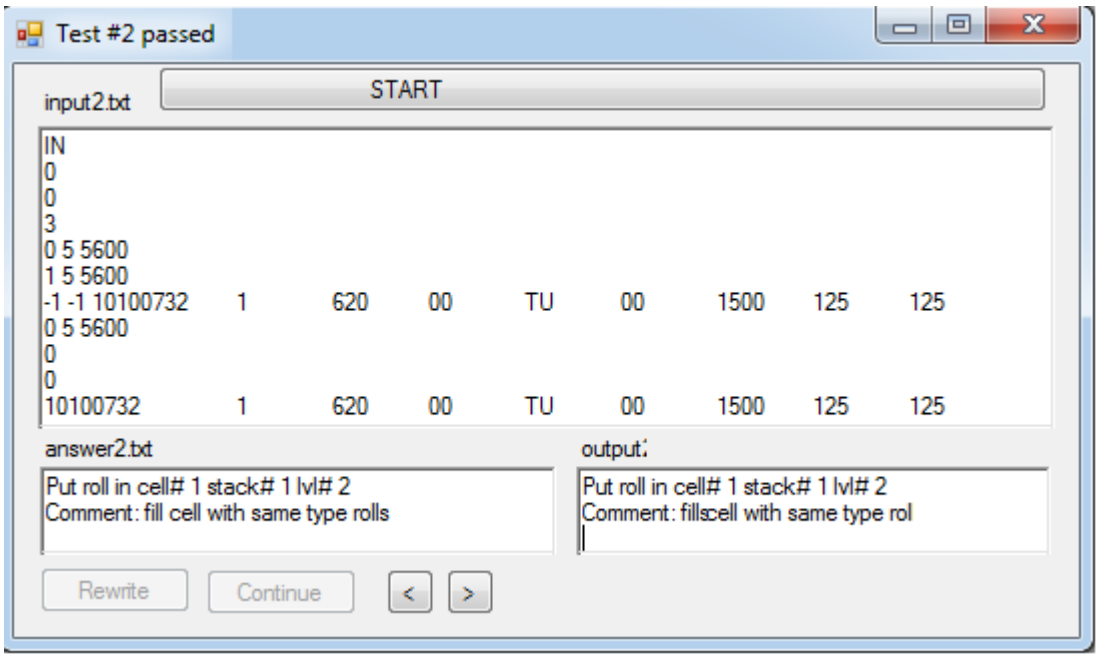

Рис. 4: скриншот утилиты для тестирования

6.3 Критерий прохождения тестов

Тест считается пройдённым, если конечный результат соответствует ожидаемому результату.

Ожидаемый результат должен присутствовать в описании каждого теста в плане тестирования.

6.4 Критерий приостановления работы

Инструментарий тестировщика был настроен таким образом, чтобы было удобно разработчику и тестировщику в одном лице. При увеличении штата есть возможность перенастроить.

Тестирование приостанавливается при первой же ошибке. После этого тестировщик должен проанализировать результат и сообщить об ошибке, либо признать новый результат не менее оптимальным, чем уже известный. После этого может быть принято решение об исправлении либо программы, либо ожидаемого результата . После достижения первой ошибки можно продолжить тестирование до следующей ошибки.

6.5 Критерий возобновления работы

После получения заявки на повторное тестирование нужно начать тестирование с самого начала.

6.6 Требуемая документация

Необходим отчёт о проведении тестирования, в нём следует указать дату, версию ПО, средства и вообще максимум информации, которая покажется тестировщику полезной.

6.7 Необходимое оборудование:

IBM-PC - совместимый компьютер или виртуальная машина с OS Windows не старше XP, .Net 4.0+.

6.8 Модульное тестирование класса Roll.

Задача класса Roll - хранить информацию о рулоне. Единственная функциональность - считывание и вывод.

Тест 1

Проверка функций read(istream &) и print(ostream &out) const

Тип: общий

Входные данные: файл с описанием рулона

Ожидаемый результат: выходной файл содержит информация о таком же рулоне, как и во входном файле

Тест 2

Проверка функций read(istream &) и print(ostream &out) const

Тип: негативный

Входные данные: файл с описанием рулона некорректен

Ожидаемый результат: выходной файл содержит информацию об ошибке

Тест 3

Проверка функций read(istream &) и print(ostream &out) const

Тип: негативный

Входные данные: входной файл отсутствует

Ожидаемый результат: выходной файл содержит информацию об ошибке

6.9 Модульное тестирование класса Cell.

Задача класса Cell - хранить информацию о ячейке. Все тесты из этого раздела будут запущены в двух вариантах: при модульном тестировании класса Cell и при интеграционном тестировании Cell + Roll. При модульном тестировании класса Cell вместо объектов класса Roll будут использоваться заглушки - числа типа int. При интеграционном тестировании класса Cell с классом Roll будут использоваться те же тесты, что и при модульном тестировании класса Cell, но при модульном номера в отчёте о проведении тестирования будут с суффиксом m, а при интеграционном - с суффиксом i.

Тест 4

Проверка функций read(istream&);

Тип: краевой

Входные данные: файл с описанием пустой ячейки

Ожидаемый результат: выходной файл содержит информация о такой же ячейке, как и во входном файле

Тест 5

Проверка функций read(istream&);

Тип: негативный

Входные данные: входной файл отсутствует

Ожидаемый результат: выходной файл содержит сообщение об ошибке

Тест 6

Проверка функций read(istream&);

Тип: негативный

Входные данные: входной файл некорректен

Ожидаемый результат: выходной файл содержит сообщение об ошибке

Тест 7

Проверка функций read(istream&);

Тип: общий

Входные данные: файл с описанием ячейки с одним рулоном

Ожидаемый результат: выходной файл содержит информация о такой же ячейке, как и во входном файле

Тест 8

Проверка функций read(istream&);

Тип: общий

Входные данные: файл с описанием ячейки с несколькими рулонами

Ожидаемый результат: выходной файл содержит информация о такой же ячейке, как и во входном файле

Тест 9

Проверка функций int rolls_cnt() const;

Тип: краевой

Входные данные: ячейка без рулонов

Ожидаемый результат: 0

Тест 10

Проверка функций int rolls_cnt() const;

Тип: общий

Входные данные: ячейка с 1 рулоном

Ожидаемый результат: 1

Тест 11

Проверка функций int rolls_cnt() const;

Тип: общий

Входные данные: ячейка с 6 рулонами

Ожидаемый результат: 6

Тест 12

Проверка функций bool is_like_last(const Roll &cur) const;

Тип: краевой

Входные данные: ячейка без рулонов

Ожидаемый результат: true

Тест 13

Проверка функций bool is_like_last(const Roll &cur) const;

Тип: общий

Входные данные: ячейка с 1 рулоном, верхний совпадает

Ожидаемый результат: true

Тест 14

Проверка функций bool is_like_last(const Roll &cur) const;

Тип: общий

Входные данные: ячейка с 6 рулонами, верхний и 3й совпадают

Ожидаемый результат: true

Тест 15

Проверка функций bool is_like_last(const Roll &cur) const;

Тип: общий

Входные данные: ячейка с 6 рулонами, верхний не совпадает

Ожидаемый результат: false

Тест 16

Проверка функций bool is_like_all(const Roll &cur) const;

Тип: краевой

Входные данные: ячейка без рулонов

Ожидаемый результат: true

Тест 17

Проверка функций bool is_like_all(const Roll &cur) const;

Тип: общий

Входные данные: ячейка с 1 рулоном, верхний совпадает

Ожидаемый результат: true

Тест 18

Проверка функций bool is_like_all(const Roll &cur) const;

Тип: общий

Входные данные: ячейка с 6 рулонами, все кроме 3го совпадают

Ожидаемый результат: false

Тест 19

Проверка функций bool is_like_all(const Roll &cur) const;

Тип: общий

Входные данные: ячейка с 6 рулонами, все совпадают

Ожидаемый результат: true

Тест 20

Проверка функции bool can_push(const Roll&, const bool force_new_stack);

Тип: краевой

Входные данные: пустая ячейка, force_new_stack = false

Ожидаемый результат: true

Тест 21

Проверка функции bool can_push(const Roll&, const bool force_new_stack);

Тип: общий

Входные данные: в ячейке частично заполнены первые 2 штабеля, новый рулон может быть помещён во 2й штабель, force_new_stack = false

Ожидаемый результат: true

Тест 22

Проверка функции bool can_push(const Roll&, const bool force_new_stack);

Тип: общий

Входные данные: в ячейке частично заполнены первые 2 штабеля, новый рулон мог быть помещён во 2й штабель, если бы был в диаметре был меньше, 3й штабель свободен , force_new_stack = false

Ожидаемый результат: false

Тест 23

Проверка функции bool can_push(const Roll&, const bool force_new_stack);

Тип: общий

Входные данные: в ячейке частично заполнены первые 2 штабеля, новый рулон мог быть помещён во 2й штабель, если бы был в диаметре был меньше, 3й штабель свободен, force_new_stack = true

Ожидаемый результат: true

Тест 24

Проверка функции bool can_push(const Roll&, const bool force_new_stack);

Тип: краевой

Входные данные: в ячейке частично заполнены первые 2 штабеля, новый рулон мог быть помещён во 2й штабель, если бы был в диаметре был меньше, в ячейке только 2 штабеля, force_new_stack = true

Ожидаемый результат: false

6.10 Модульное тестирование подсистемы вывода.

Тест 25

Проверка вывода ответа

Тип: общий

Входные данные: подсистеме вывода подаётся задача вывести ответ - ячейка №2

Ожидаемый результат: в выходном файле записано, что рулон следует разместить во 2й ячейке

Тест 26

Проверка вывода ответа

Тип: общий

Входные данные: подсистеме вывода подаётся задача вывести ответ - ячейка №3

Ожидаемый результат: в выходном файле записано, что рулон следует разместить в 3й ячейке

6.11 Интеграционное тестирование.

Интеграционное тестирование классов Cell+Roll описано выше в разделе "Тестирование класса Cell".

6.12 Системное тестирование.

Системное тестирование проверяет работу системы в целом. Подаётся файл с состоянием склада и новым рулоном, результат - ячейка, в которой следует разместить новый рулон. В описании ячеек будет список рулонов, в котором разные буквы означают разные рулоны, а одинаковые буквы - одинаковые рулоны. Конкретные списки свойств рулонов указываться не будут. На этих же тестах будет производиться модульное тестирование подсистемы решения. Для модульного тестирования системы решения в отчёте о проведении тестирования в номерах тестов будет указан суффикс m, для системного тестирования - суффикс s.

Тест 27

Проверка того, что хотя бы что-то работает

Тип: системный
Входные данные: склад, на котором 2 ячейки.
1я ячейка: A
2я ячейка: B
Новый рулон: типа B

Ожидаемый результат: поместить во 2ю ячейку

Тест 28

Проверка того, что одинаковые рулоны собираются вместе

Тип: системный
Входные данные: склад, на котором 3 ячейки.
1я ячейка: CBA
2я ячейка: ABA
3я ячейка: BCA
Новый рулон: типа A

Ожидаемый результат: поместить во 2ю ячейку

Тест 29

Проверка того, что если свободных ячеек мало, они не занимаются

Тип: системный
Входные данные: склад, на котором 100 ячеек, в 98 из них по одному рулону (между собой как одинаковые, так и разные)

Новый рулон: отличается от всех

Ожидаемый результат: поместить в одну из занятых ячеек

Тест 30

Проверка того, что если свободных ячеек много, они занимаются

Тип: системный
Входные данные: склад, на котором 100 ячеек, в 89 из них по одному рулону (между собой как одинаковые, так и разные)
Новый рулон: отличается от всех

Ожидаемый результат: поместить в одну из свободных ячеек

Тест 31

Проверка того, что если свободных ячеек мало, но новый рулон вместе с однотипными рулонами из плана выработки может целиком занять пустую ячейку

Тип: системный
Входные данные: склад, на котором 100 ячеек, в 98 из них по одному рулону (между собой как одинаковые, так и разные)
Новый рулон: отличается от всех
План выработки: 100 рулонов, однотипны с новым

Ожидаемый результат: поместить в одну из свободных ячеек

Тест 32

Проверка того, что если есть ячейка-свалка рулонов разных типов, а есть ячейки в которых рулоны хранятся упорядоченно, такая ситуация сохраняется

Тип: системный
Входные данные: склад, на котором 5 ячеек.
1я ячейка: CCCC
2я ячейка: D
3я ячейка: BGBTK
4я ячейка: A
5я ячейка: EEFF

Новый рулон: типа Z

Ожидаемый результат: поместить в Зю ячейку

6.13 Требования к тестировщикам

Для тестирования достаточно одного тестировщика, от него требуется лишь немножко ума и сообразительности, т.к. запуск утилиты для тестирования очень прост.

7 Пример исходного кода

```
private void startTesting(bool oneTest)
{
	rewriteB.Enabled = continueB.Enabled = false;
	for (; ; testNumber++)
	{
		int res = readAnsOut(testNumber);
		if (0 == res)
		{
		MessageBox.Show("All of " + (testNumber - 1) + " tests have been passed");
			return;
		}
		else if (1 == res)
			return;
		bool isEqual = checkAnsOut();
		if (oneTest || !isEqual)
			updateTextBoxes(testNumber);
		if (!isEqual)
		{
			rewriteB.Enabled = continueB.Enabled = true;
			return;
		}
		this.Text = "Test #" + testNumber + " passed";
		if (oneTest)
			return;
	}
}
```

8. Методы покрытия

Расчёт покрытия тестами относительно исполняемого кода производится по формуле:

Covering = tested_length / code_length

tested_length - количество строк кода, покрытых тестами

code_length - общее количество строк кода в программе

tested_length = 947

code_length = 1179

Covering = 947/1179=80,32%

9. Отчёт о проведении тестирования

Тестирование проводится при помощи самодельной утилиты.

№ теста	Дата	Результат
1	15-01-2015	ОК
2	15-01-2015	ОК
3	15-01-2015	ОК
4m	15-01-2015	ОК
5m	15-01-2015	ОК
6m	15-01-2015	ОК
7m	15-01-2015	ОК
8m	15-01-2015	ОК
9m	15-01-2015	ОК
10m	15-01-2015	ОК
11m	15-01-2015	Runtime error (Ошибка № 1)
12m	15-01-2015	ОК
13m	15-01-2015	ОК
14m	15-01-2015	ОК
15m	15-01-2015	ОК
16m	15-01-2015	ОК
17m	15-01-2015	ОК
18m	15-01-2015	ОК
19m	15-01-2015	ОК
20m	15-01-2015	ОК
21m	15-01-2015	ОК

22m	15-01-2015	OK
23m	15-01-2015	OK
24m	15-01-2015	OK
4i	15-01-2015	OK
5i	15-01-2015	OK
6i	15-01-2015	OK
7i	15-01-2015	OK
8i	15-01-2015	OK
9i	15-01-2015	OK
10i	15-01-2015	OK
11i	15-01-2015	Runtime error (Ошибка № 1)
12i	15-01-2015	OK
13i	15-01-2015	OK
14i	15-01-2015	OK
15i	15-01-2015	OK
16i	15-01-2015	OK
17i	15-01-2015	OK
18i	15-01-2015	OK
19i	15-01-2015	OK
20i	15-01-2015	OK
21i	15-01-2015	OK
22i	15-01-2015	OK
23i	15-01-2015	OK
24i	15-01-2015	OK
25	15-01-2015	OK
26	15-01-2015	OK
27m	15-01-2015	OK

28m	15-01-2015	OK
29m	15-01-2015	OK
30m	15-01-2015	OK
31m	15-01-2015	OK
32m	15-01-2015	OK
27s	15-01-2015	OK
28s	15-01-2015	OK
29s	15-01-2015	OK
30s	15-01-2015	OK
31s	15-01-2015	OK
32s	15-01-2015	OK

10. Отчёт об ошибке

№ ошибки: 1

№ теста: 11i

Программа была протестирована под ОС Windows 7, установлен .Net 4.0. Версия программы - от 5 января 2015. Тест для более ранних версий не использовался, т.к. этого теста раньше не было.

Алгоритм воспроизведения ошибки:

1. Создаём объект класса Cell, в котором нет рулонов

2. Создаём произвольный рулон cur. В тесте использован рулон с параметрами "10100732 1 620 00 B 00 1500 125 125", но баг воспроизводится и при других параметрах.

3. Вызываем у ячейки метод "is_like_last(cur)"

4. Составить отчёт о том, воспроизведена ошибка, или нет

11. Текущие результаты

Разработана программа, реализующая все требуемые функции и удовлетворяющая всем требованиям

Программа успешно прошла тестирование.