**Валидация, Верификация и План за тестване**

*Реализиране на алгоритъма “Frequent Pattern Growth”*

**Revision Sheet**

|  |  |  |
| --- | --- | --- |
| **Release No.** | **Date** | **Revision Description** |
| Рев. 0 | 29.08.20г. | Преглед и планиране на корекции по шаблона за Тест план |
| Рев. 1 | 30.08.20г. | Добавяне на тестове на Черни кутии |
| Рев. 2 | 08.09.20г. | Добавяне на тестове на Бели кутии |
| Рев. 3 | 13.09.20г. | Финализиране на Тест плана |
|  |  |  |
|  |  |  |
|  |  |  |

**Валидация, Верификация и Тестов план**

**СЪДЪРЖАНИЕ**

Страница #

1.0 Обща информация 2-1

1.1 Цел 2-1

1.2 Обхват 2-1

1.3 Преглед на системата 2-1

2.0 Оценяване на теста 2-1

2.1 Критерии за оценка на теста 2-1

2.2 Критерии за приемане от потребителската система 2-1

3.0 График на тестване 3-1

3.1 Цялостен график на тестване 3-1

3.2 Място на извършване на теста 3-1

3.2.1 Изисквания към оборудването 3-1

3.2.2 Изисквания към софтуера 3-1

3.2.3 Изисквания към персонала 3-1

3.2.4 Инструменти за тестване 3-1

4.0 Характеристики на тестването 4-1

4.1 Условия на тестването 4-1

4.2 Дълбочина на тестването 4-1

4.3 Ограничения при тестването 4-1

4.4 Напредък при тестването 4-1

5.0 Описание на теста 5-2

5.1 WhiteBox тест1 5-2

5.1.1 Системни функции 5-2

5.1.2 Връзка тест-функция 5-2

5.1.3 Средства за контрол 5-2

5.1.4 Данни за тестване 5-2

5.1.4.1 Входни данни 5-2

5.1.4.2 Входни команди 5-2

5.1.4.3 Изходни данни 5-2

5.1.4.4 Известяване при изход 5-2

5.2 WhiteBox тест2 5-2

5.2.1 Системни функции 5-2

5.2.2 Връзка тест-функция 5-3

5.2.3 Средства за контрол 5-3

5.2.4 Данни за тестване 5-3

5.2.4.1 Входни данни 5-3

5.2.4.2 Входни команди 5-3

5.2.4.3 Изходни данни 5-4

5.2.4.4 Известяване при изход 5-4

5.3 WhiteBox тест3 5-3

5.3.1 Системни функции 5-3

5.3.2 Връзка тест-функция 5-3

5.3.3 Средства за контрол 5-3

5.3.4 Данни за тестване 5-3

5.3.4.1 Входни данни 5-3

5.3.4.2 Входни команди 5-3

5.3.4.3 Изходни данни 5-4

5.3.4.4 Известяване при изход 5-4

5.4 BlackBox тест1 5-3

5.4.1 Системни функции 5-3

5.4.2 Връзка тест-функция 5-3

5.4.3 Средства за контрол 5-3

5.4.4 Данни за тестване 5-3

5.4.4.1 Входни данни 5-3

5.4.4.2 Входни команди 5-3

5.4.4.3 Изходни данни 5-4

5.4.4.4 Известяване при изход 5-4

5.5 BlackBox тест2 5-3

5.5.1 Системни функции 5-3

5.5.2 Връзка тест-функция 5-3

5.5.3 Средства за контрол 5-3

5.5.4 Данни за тестване 5-3

5.5.4.1 Входни данни 5-3

5.5.4.2 Входни команди 5-3

5.5.4.3 Изходни данни 5-4

5.5.4.4 Известяване при изход 5-4

# Обща информация

## 1.1 Цел

Целта на системата е да може да възпроизведе тестове, които да покажат дали алгоритъма “Frequent Pattern” работи правилно.

## 1.2 Обхват

Тестовият план ще обхване част от кода на проекта, фокусирайки се върху фрагментите свързани с обработка на входно/изходните данни подадени от файл.

## 1.3 Преглед на системата

Системата представлява среда за разработка Microsoft Visual Studio с програмен език С++ и набор от инструменти базирани на open-source - CMake

# Оценяване на теста

## 2.1 Критерии за оценка на теста

Очаква се системата да може да приеме и обработи данните в разумно време(по-малко от 5 секунди за файловете, които тестват алгоритъма без реалните данни и около 5 минути за обработката на файла с реалните данни). Също така от системата се очаква да изнесе резултатът в четим за един обикновен потребител вид на екрана.

## 2.2 Критерии за приемане от потребителската система

Реагиране в рамките на зададения прозорец време при изпълнение на операция. Срив на програмата не води до загуба на входните данни и/или до срив на системата. Лесен и удобен за ползване интерфейс за работа. Съобщенията за грешки да са приведени в разбираем за тестиращия вид.

# график на тестване

## 3.1 Цялостен график на тестване

Тестовете започват след установяване на системата в готовност и индивидуално е редно да имат времетраене от <1минута до >5 минути на тест (5 или повече минути в случая на четенето и обработката на файла, съдържащ реални данни). Наличен е един тестер, който среди за правилната работа на програмата.

Готовността на системата се изразява в това програмата да е пусната и да не е забила, правилните входни данни да са в готовност, да няма други тежки процеси в изпълнение в момента на ползване на програмата.

## 3.2 Място на извършване на теста

Мястото на извършване на тестовете е в която и да е лаборатория, сграда или стая снабдена с подходяща за работа базова среда.

### 3.2.1 Изисквания към оборудването

Работещ компютър (настолен или преносим). Връзката с интернет, по възможност. Наличие на електрозахранване и подходяща мебел за приютяване на компютърната конфигурация.

### 3.2.2 Изисквания към софтуера

Да има Microsoft Visual Studio с необходимите С++ пакети и инсталиран CMake

### 3.2.3 Изисквания към персонала

Необходим е само един тестер, който притежава нужните познания за писане и проверка на дадения програмен език.

Тестерът има задачата да подготви средата и да пусне тестовете, системата автоматично ще изведе резултатите.

### 3.2.4 Инструменти за тестване

Вградени в средата за пускане на кода.

# характеристики на тестване

## 4.1 Условия на тестване

Необходимо е ползването на контролни данни за проверка на поведението на програмата (FPTest.cpp и PhilippeFourierExampleTest.cpp). След което ще се пристъпи към ползване на реалните данни (VVPSTest.cpp).

## 4.2 Дълбочина на тестване

Тъй като самата програма не е толкова обемна в сравнение със стандартен проект в софтуерна компания, няма да е необходимо прекалено задълбочаване в тестването на поведението.

## 4.3 Ограничения при тестването

Главно ограниченията ще идват от производителността на ползваната система, версията на софтуера и обема на данните предназначени за обработка.

## 4.4 Напредък при тестването

Има се впредвид последователност на тестовете. Не се очаква да се наблюдава свързаност между различните тестове(вход/изход от един тест да преминава във вход/изход на друг).

# описание на теста

## 5.1 WhiteBox тест1

//наименование на теста и описание на самия тест

Файл VVPSTest.cpp

Този файл е основният за показване за това как работи алгоритъма в реална среда. В него е зададена само една тестова група с общо 12 теста.

Теста се осъществява чрез директно пускане на програмата FrequentPatternGrowth\_test.exe от Visual Studio.  
 Първи тест на тип “Бяла кутия” е група VVPSTestSuit с тестове от Test1 до Test12.

### 5.1.1 Системни функции

//кои системни функции и функции за комуникация подлежат на проверка

Функциите, които ще бъдат проверени са -

TEST(VVPSTestSuit, TestPattern1){} до TEST(VVPSTestSuit, TestPattern12){}

### 5.1.2 Връзка тест-функция

//какви функции ползваме за тестовете и как това влияе на тях

За този тест се използват Функциите -   
TEST(VVPSTestSuit, TestPattern1){} до TEST(VVPSTestSuit, TestPattern12){}

### 5.1.3 Средства за контрол

//по какъв начин се контролира теста(ръчно,полуавтоматично,автоматично)

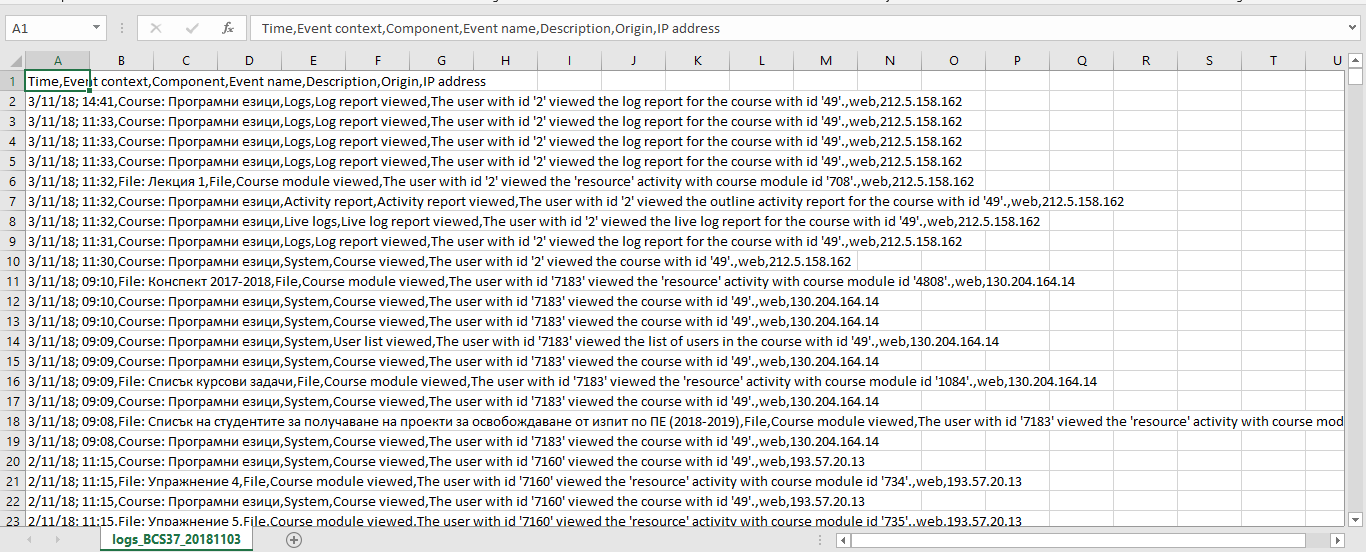
Теста бива контролиран автоматично, Чрез стартиране на FrequentPatternGrowth\_test.exe , никаква друга намеса от страна на тестващият не е необходима.

### 5.1.4 Данни за тестване

#### 5.1.4.1 Входни данни

//описание на входните данни ползвани за теста

Входните данни са представени като .csv файл (Файл, който използва запетая за разделяне на стойности. Всеки ред на файла е запис на данни), файла съдържа общо 63774 реда с реална информация



#### 5.1.4.2 Входни команди

//команди за манипулация на теста

Използваните команди/функции биват от вида на Google Test “Очакваният резултат е равенство” (EXPECTED\_EQ) при зададена команда за преброяване на елементите от всички транзакции, където се задават параметрите – {Търсен елемент, сравнение със зададена стойност} и двоично представяне дали сравнението е очакваното или не ( 1 или 0 ).

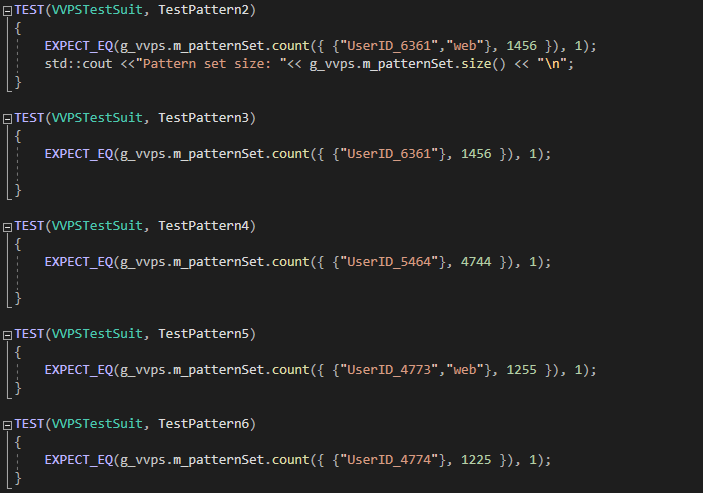
Пример

TEST(VVPSTestSuit, TestPattern3)

{

EXPECT\_EQ(g\_vvps.m\_patternSet.count({ {"UserID\_6361"}, 1456 }), 1);

}



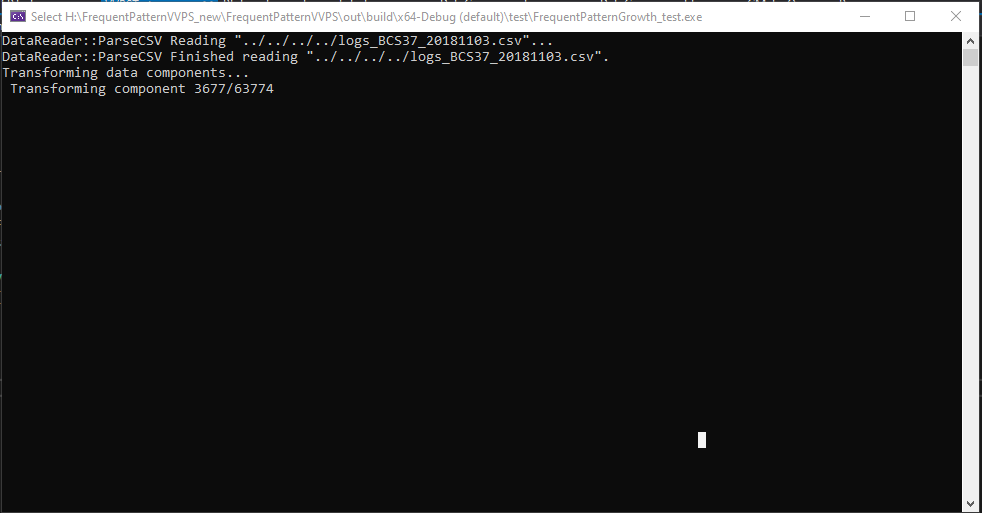
#### 5.1.4.3 Изходни данни

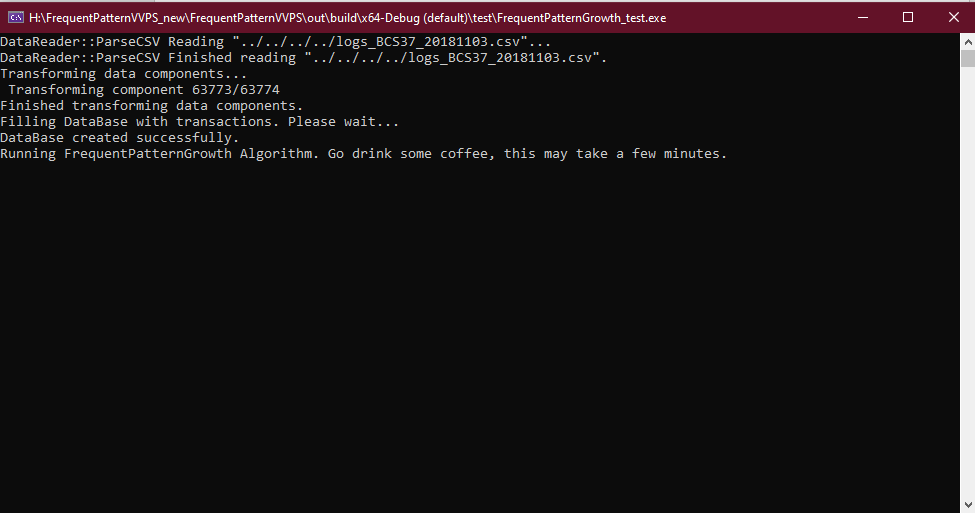
//къде и как се появяват изходните данни, как програмата определя че има изходни данни, има ли //разлика между очаквания резултат и получения резултат

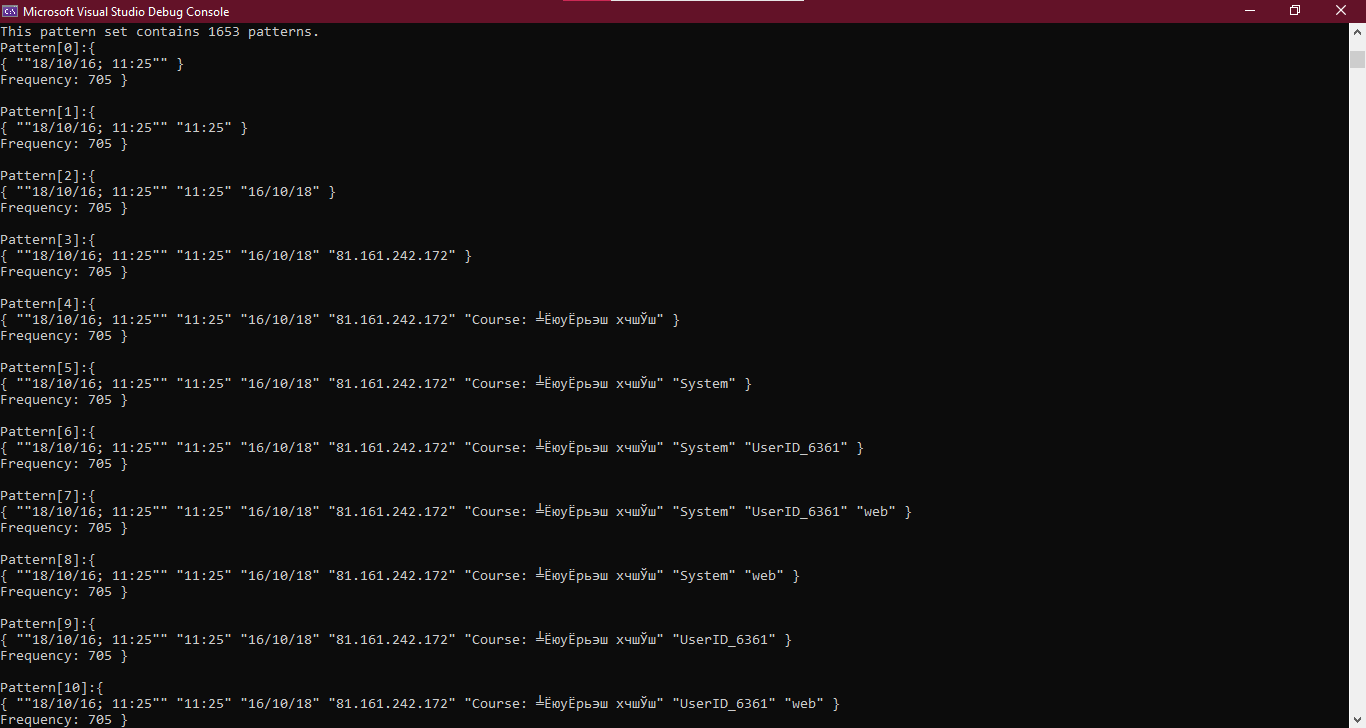
Изходните данни биват представени в конзолен прозорец под формата на текст.

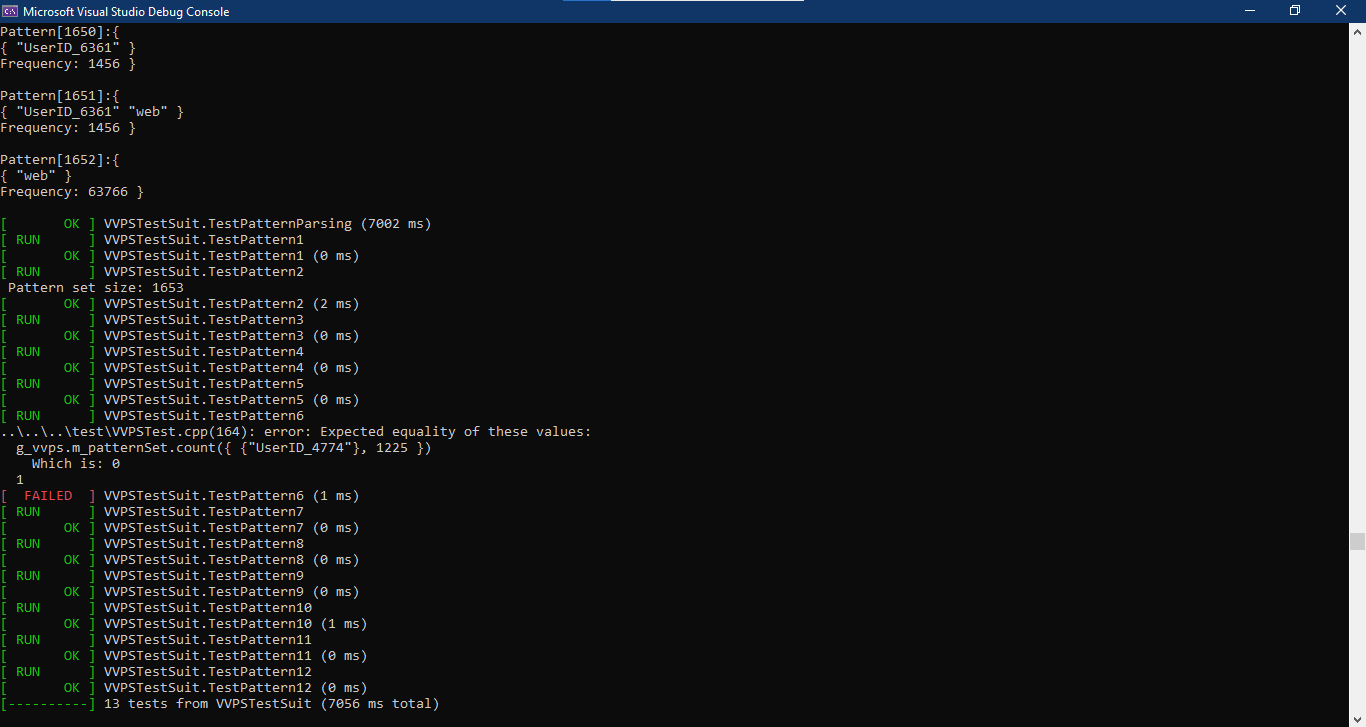
За наше улеснение и сравнение в работата на алгоритъма, цялата форма на извършеният алгоритъм бива представена текстово – Първо се отваря желаният файл “logs\_BCS37\_20181103.csv”, след това се обработва 1вата запетайка на елемента между датата и часа на всеки ред от файла, за да бъде четима датата според изискването за “Time”. В снимка 1 се виждат трансформиращите компоненти (Редове) – а именно 63774 реда.

Очакваният резултат е всеки без 6-ти тест да е успешен.

(Снимка 1 )



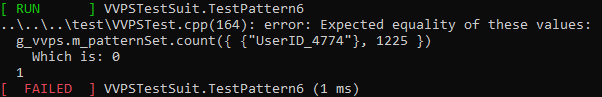




#### 5.1.4.4 Известяване при изход

//описване на начина, по който се уведомява потребителя за изходните данни

Изходните данни биват представени по лесно разбираем начин – Първоначално бива определено за даден тест нали е стартирал, чрез оцветено в зелено думата “RUN” и ако теста не е преминал успешно, той бива маркиран на следващия ред в оцветена в червено думата “FAILED”, с допълнителна информация за проблема предхождаща го.



В този случай бива тествано дали даденият UserID\_4774 присъства в алгоритъма 1225 пъти, и теста правилно се проваля, защото се среща по-малко пъти и не влиза в минимално зададения праг от 700.

## 5.2 WhiteBox тест2

//наименование на теста и описание на самия тест

Файл VVPSTest.cpp

Този файл е основният за показване за това как работи алгоритъма в реална среда. В него е зададена само една тестова група с общо 12 теста.

Теста се осъществява чрез директно пускане на програмата FrequentPatternGrowth\_test.exe от Visual Studio.  
 Втори тест на тип “Бяла кутия” е група VVPSTestSuit с тестове от Test1 до Test12.

### 5.2.1 Системни функции

//кои системни функции и функции за комуникация подлежат на проверка

Функциите, които ще бъдат проверени са -

TEST(VVPSTestSuit, TestPattern1){} до TEST(VVPSTestSuit, TestPattern12){}

### 5.2.2 Връзка тест-функция

//какви функции ползваме за тестовете и как това влияе на тях

За този тест се използват Функциите -   
TEST(VVPSTestSuit, TestPattern1){} до TEST(VVPSTestSuit, TestPattern12){}

### 5.2.3 Средства за контрол

//по какъв начин се контролира теста(ръчно,полуавтоматично,автоматично)

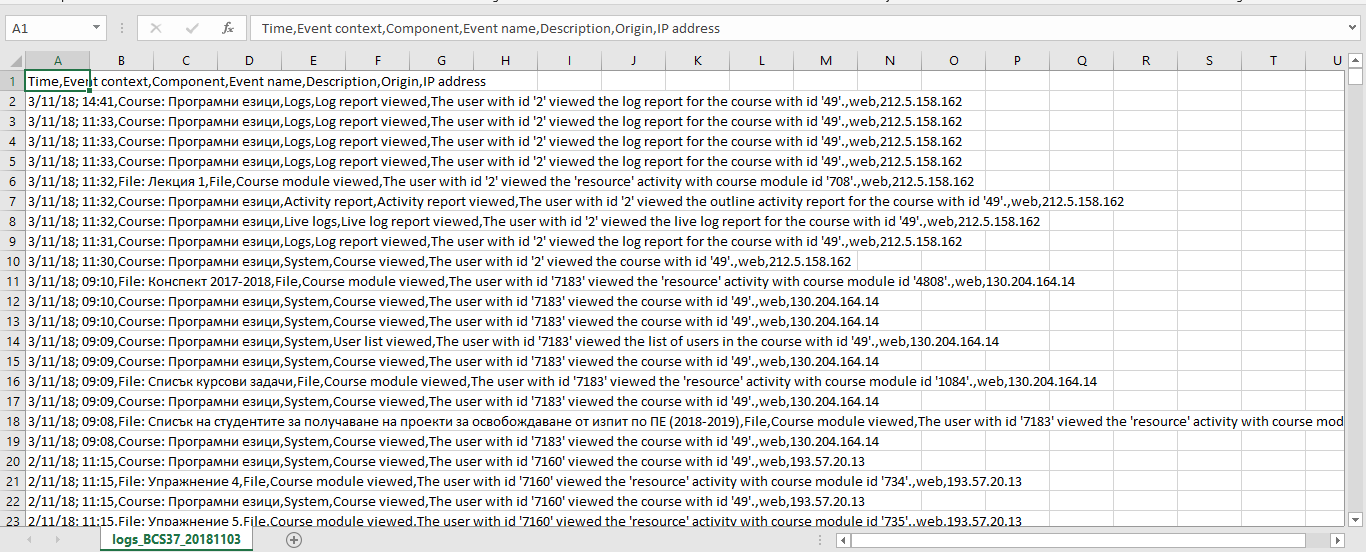
Теста бива контролиран автоматично, Чрез стартиране на FrequentPatternGrowth\_test.exe , никаква друга намеса от страна на тестващият не е необходима.

### 5.2.4 Данни за тестване

#### 5.2.4.1 Входни данни

//описание на входните данни ползвани за теста

Входните данни са представени като .csv файл (Файл, който използва запетая за разделяне на стойности. Всеки ред на файла е запис на данни), файла съдържа общо 63774 реда с реална информация



#### 5.2.4.2 Входни команди

//команди за манипулация на теста

Използваните команди/функции биват от вида на Google Test “Очакваният резултат е равенство” (EXPECTED\_EQ) при зададена команда за преброяване на елементите от всички транзакции, където се задават параметрите – {Търсен елемент, сравнение със зададена стойност} и двоично представяне дали сравнението е очакваното или не ( 1 или 0 ).

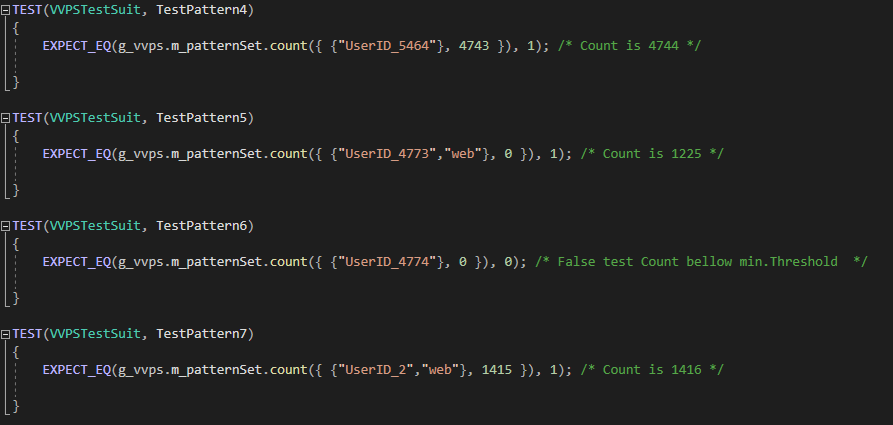
Пример

TEST(VVPSTestSuit, TestPattern3)

{

EXPECT\_EQ(g\_vvps.m\_patternSet.count({ {"UserID\_6361"}, 1456 }), 1);

}



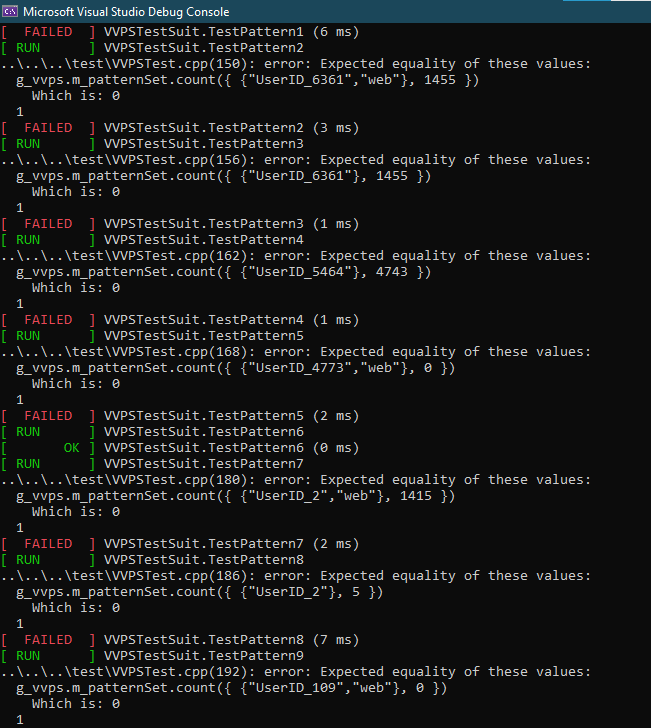
#### 5.2.4.3 Изходни данни

//къде и как се появяват изходните данни, как програмата определя че има изходни данни, има ли //разлика между очаквания резултат и получения резултат

Изходните данни биват представени в конзолен прозорец под формата на текст.

За наше улеснение и сравнение в работата на алгоритъма, цялата форма на извършеният алгоритъм бива представена текстово – Първо се отваря желаният файл “logs\_BCS37\_20181103.csv”, след това се обработва 1вата запетайка на елемента между датата и часа на всеки ред от файла, за да бъде четима датата според изискването за “Time”, а също и конфигурацията на UserID и CourseID. В следващата снимка се виждат провалените тестове без 6-ти, поради спецификата на теста да провери дали липсва такъв резултат.

Очакваният резултат е само 6-ти да бъде успешен.



#### 5.2.4.4 Известяване при изход

//описване на начина, по който се уведомява потребителя за изходните данни

Изходните данни биват представени по лесно разбираем начин – Първоначално бива определено за даден тест нали е стартирал, чрез оцветено в зелено думата “RUN” и ако теста е преминал успешно, той бива маркиран на следващия ред в оцветена в зелено думата “OK”.



В този случай бива тествано дали даденият UserID\_4774 **НЕ** се среща 0 пъти, теста е положителен, защото търсеният елемент не преминава границата от 700 и не бива записан в сета. Не съществува.

## 5.3 WhiteBox тест3

//наименование на теста и описание на самия тест

Файл VVPSTest.cpp

Този файл е основният за показване за това как работи алгоритъма в реална среда. В него е зададена само една тестова група с общо 12 теста.

Теста се осъществява чрез директно пускане на програмата FrequentPatternGrowth\_test.exe от Visual Studio.  
 Трети тест на тип “Бяла кутия” е група VVPSTestSuit е тест дали ще отвори несъществуващ файл.

### 5.3.1 Системни функции

//кои системни функции и функции за комуникация подлежат на проверка

Функциите, които ще бъдат проверени са -

TEST(VVPSTestSuit, TestPattern1){} до TEST(VVPSTestSuit, TestPattern12){}

### 5.3.2 Връзка тест-функция

//какви функции ползваме за тестовете и как това влияе на тях

За този тест се използват Функциите -   
TEST(VVPSTestSuit, TestPattern1){} до TEST(VVPSTestSuit, TestPattern12){}

### 5.3.3 Средства за контрол

//по какъв начин се контролира теста(ръчно,полуавтоматично,автоматично)

Теста бива контролиран автоматично, Чрез стартиране на FrequentPatternGrowth\_test.exe , никаква друга намеса от страна на тестващият не е необходима.

### 5.3.4 Данни за тестване

#### 5.3.4.1 Входни данни

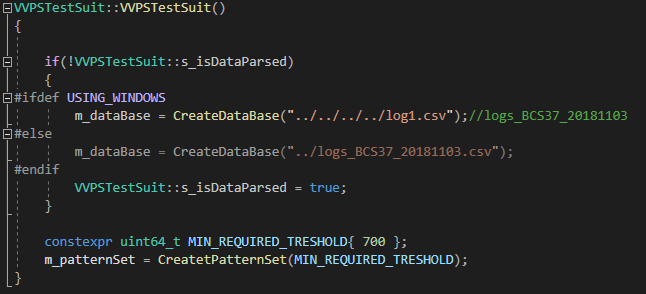
//описание на входните данни ползвани за теста

В този случай, няма да има входни данни, поради липсата на файла

#### 5.3.4.2 Входни команди

//команди за манипулация на теста

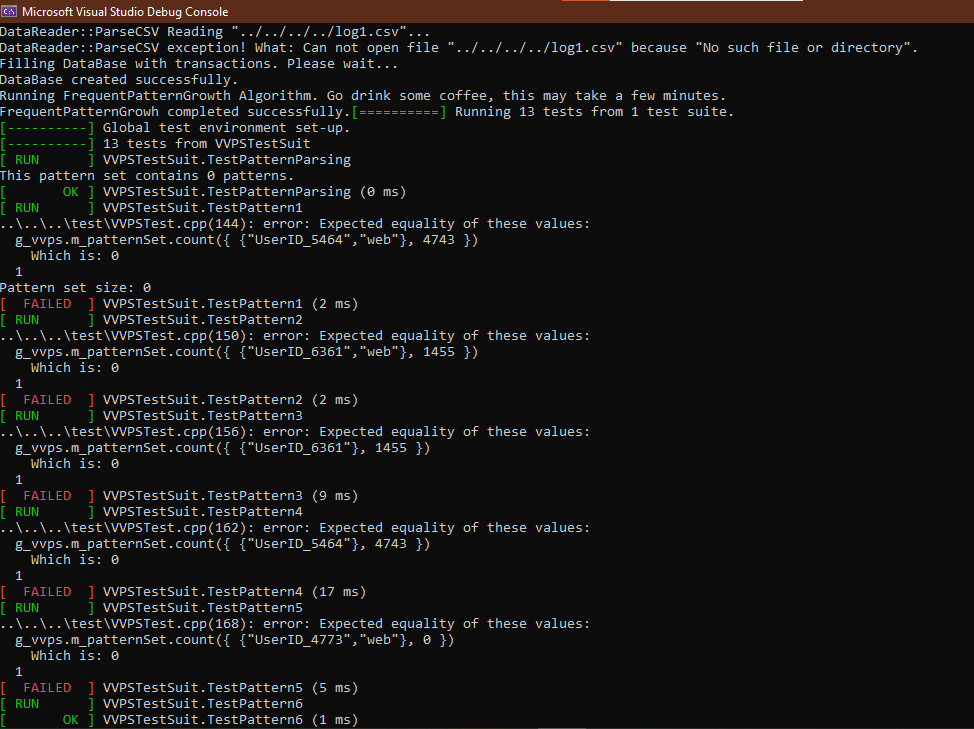
Манипулацията бива само в промяна на търсенето на файла “log1” (несъществуващ).



#### 5.3.4.3 Изходни данни

//къде и как се появяват изходните данни, как програмата определя че има изходни данни, има ли //разлика между очаквания резултат и получения резултат

Очакваният резултат е както следва – изписването на липсата на файла, но с последващи провалени тестове, освен тест 6, който тества за липсата на дадена информация.



#### 5.3.4.4 Известяване при изход

//описване на начина, по който се уведомява потребителя за изходните данни



(Снимка 1)

Тук програмата ни известява, че даденият файл не е намерен.



(Снимка 2)

Въпреки това, теста от преди пак преминава успешно – защото както тогава, така и сега, теста търси за липсата на информация, която просто не съществува.

## 5.4 BlackBox тест1

//наименование на теста и описание на самия тест

Файл FPTest.cpp

Този файл е примерна постановка за това как работи алгоритъма. В него са зададени 3 тестови групи.   
 Теста се осъществява чрез директно пускане на програмата FrequentPatternGrowth\_test.exe от Visual Studio.   
 Първи тест на тип “Черна кутия” е група FPGrowthPatternGroup1 с тестове от Test1 до Test20

### 5.4.1 Системни функции

//кои системни функции и функции за комуникация подлежат на проверка

Функция, която ще бъде проверена е

const FP::DataBase transactions{

{ a, b },

{ b, c, d },

{ a, c, d, e },

{ a, d, e },

{ a, b, c },

{ a, b, c, d },

{ a },

{ a, b, c },

{ a, b, d },

{ b, c, e },

};

### 5.4.2 Връзка тест-функция

//какви функции ползваме за тестовете и как това влияе на тях

За този тест се използват Функциите -   
TEST\_F(FPGrowthPatternGroup1, Test1) до TEST\_F(FPGrowthPatternGroup1, Test20)

### 5.4.3 Средства за контрол

//по какъв начин се контролира теста(ръчно,полуавтоматично,автоматично)

Теста бива контролиран автоматично, Чрез стартиране на FrequentPatternGrowth\_test.exe , никаква друга намеса от страна на тестващият не е необходима.

### 5.4.4 Данни за тестване

#### 5.4.4.1 Входни данни

//описание на входните данни ползвани за теста

Входните данни са във вид на транзакции съставени от няколко символа във вид на буквени означения от латинската азбука ( от а до e ).

const FP::DataBase transactions{

{ a, b },

{ b, c, d },

{ a, c, d, e },

{ a, d, e },

{ a, b, c },

{ a, b, c, d },

{ a },

{ a, b, c },

{ a, b, d },

{ b, c, e },

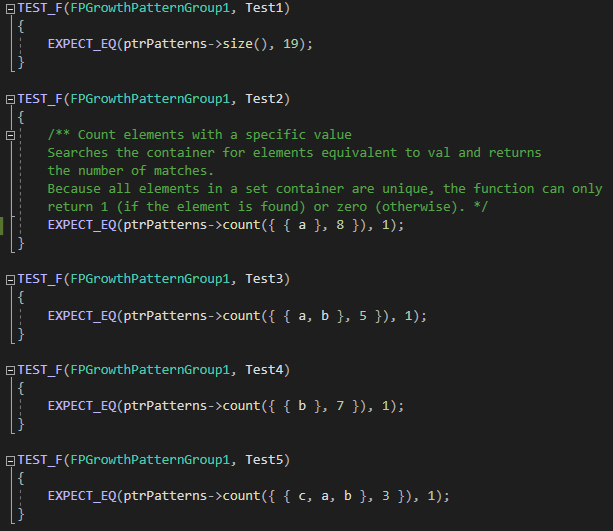
};

#### 5.4.4.2 Входни команди

//команди за манипулация на теста

Използваните команди/функции биват от вида “Очакваният резултат е равенство” при зададена команда за преброяване на символите от всички транзакции, където се задават параметрите – Търсен символ, сравнение със зададена стойност и двоично представяне дали сравнението е очакваното или не ( 1 или 0 ).

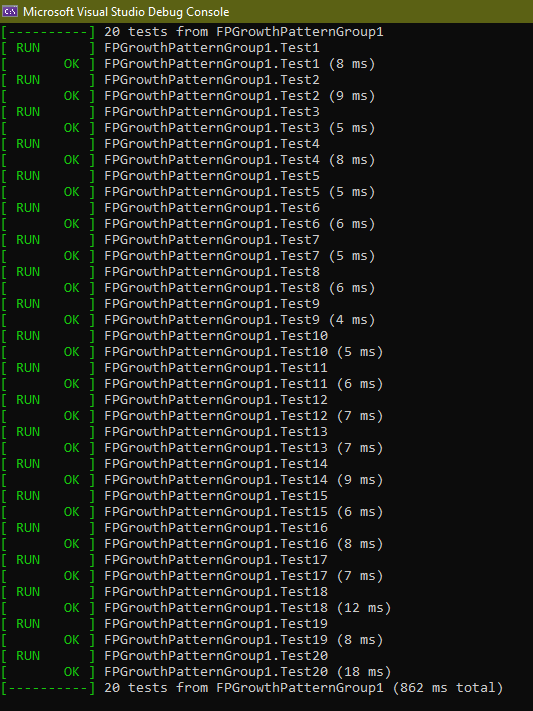
Пр. EXPECT\_EQ(ptrPatterns->count({ { a }, 8 }), 1);



#### 5.4.4.3 Изходни данни

//къде и как се появяват изходните данни, как програмата определя че има изходни данни, има ли //разлика между очаквания резултат и получения резултат

Изходните данни се появяват веднага след стартиране на теста във формата представена от библиотеката “Google Test”, в отворен нов конзолен прозорец, който се запълва постепенно със резултатите от сравненията.



#### 5.4.4.4 Известяване при изход

//описване на начина, по който се уведомява потребителя за изходните данни

Изходните данни биват представени по лесно разбираем начин – Първоначално бива определено за даден тест нали е стартирал, чрез оцветено в зелено думата “RUN” и ако теста е преминат успешно, той бива маркиран на следващия ред в оцветена в зелено думата “OK”.



При този тест, всички тестове преминаха успешно.

## 5.5 BlackBox тест2

//наименование на теста и описание на самия тест

Файл FPTest.cpp

Този файл е примерна постановка за това как работи алгоритъма. В него са зададени 3 тестови групи.   
 Теста се осъществява чрез директно пускане на програмата FrequentPatternGrowth\_test.exe от Visual Studio.   
 Втори тест на тип “Черна кутия” е група FPGrowthPatternGroup2 с тестове от Test1 до Test20

### 5.5.1 Системни функции

//кои системни функции и функции за комуникация подлежат на проверка

Функцията, която подлежи на проверка е

const FP::DataBase transactions{

{ a, b, d, e },

{ b, c, e },

{ a, b, d, e },

{ a, b, c, e },

{ a, b, c, d, e },

{ b, c, d },

};

### 5.5.2 Връзка тест-функция

//какви функции ползваме за тестовете и как това влияе на тях

За този тест се използват Функциите -   
TEST\_F(FPGrowthPatternGroup1, Test1) до TEST\_F(FPGrowthPatternGroup1, Test20)

### 5.5.3 Средства за контрол

//по какъв начин се контролира теста(ръчно,полуавтоматично,автоматично)

Теста бива контролиран автоматично, Чрез стартиране на FrequentPatternGrowth\_test.exe , никаква друга намеса от страна на тестващият не е необходима.

### 5.5.4 Данни за тестване

#### 5.5.4.1 Входни данни

//описание на входните данни ползвани за теста

Входните данни са във вид на транзакции съставени от няколко символа във вид на буквени означения от латинската азбука ( от а до e ).

const FP::DataBase transactions{

{ a, b, d, e },

{ b, c, e },

{ a, b, d, e },

{ a, b, c, e },

{ a, b, c, d, e },

{ b, c, d },

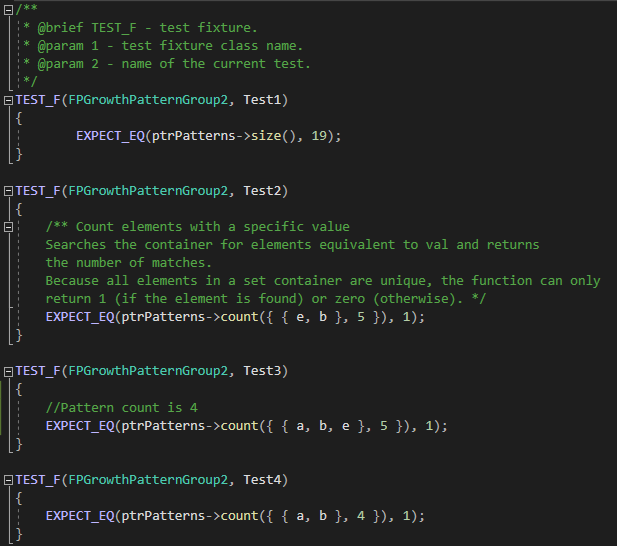
};

#### 5.5.4.2 Входни команди

//команди за манипулация на теста

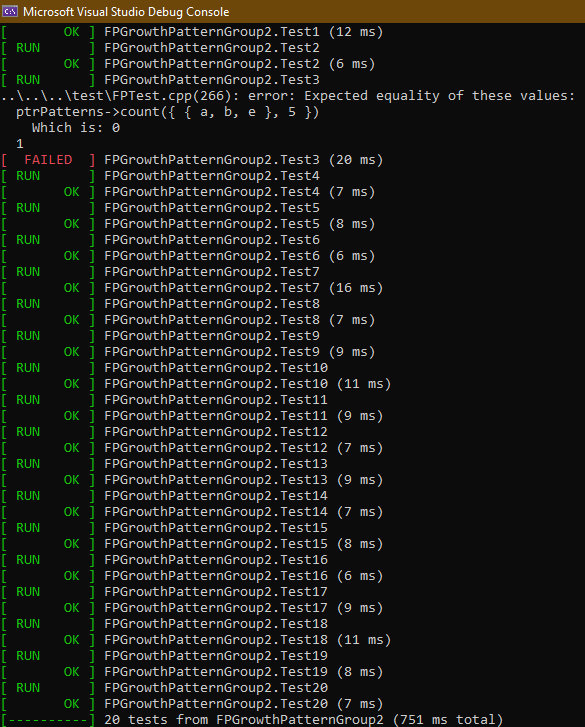
Използваните команди/функции биват от вида “Очакваният резултат е равенство” при зададена команда за преброяване на символите от всички транзакции, където се задават параметрите – Търсен символ, сравнение със зададена стойност и двоично представяне дали сравнението е очакваното или не ( 1 или 0 ).

Пр. EXPECT\_EQ(ptrPatterns->count({ { a }, 8 }), 1);



#### 5.5.4.3 Изходни данни

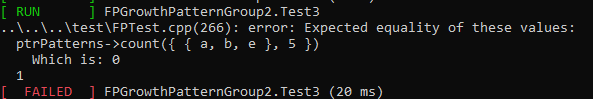
//къде и как се появяват изходните данни, как програмата определя че има изходни данни, има ли //разлика между очаквания резултат и получения резултат



#### 5.5.4.4 Известяване при изход

//описване на начина, по който се уведомява потребителя за изходните данни

Изходните данни биват представени по лесно разбираем начин – Първоначално бива определено за даден тест нали е стартирал, чрез оцветено в зелено думата “RUN” и ако теста не е преминал успешно, той бива маркиран на следващия ред в оцветена в червено думата “FAILED”, с допълнителна информация за проблема предхождаща го.



В този случай има нарочно провален тест, който да докаже правилната работа на алгоритъма.