Guion de prácticas

Shopping4

EventSet e Index en el Heap

Abril de 2021

UNIVERSIDAD DE GRANADA

ETSIIT Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

DECSAI

Metodología de la Programación

DGIM-GII-GADE

Curso 2020/2021

Índice

Registro de actividad

Date Time	Event Type	Product ID	Category ID	Category Code	Brand	Price	User ID	Session ID
string	string	string	string	string	string	double	string	string

Figura 1: Estructura del registro de actividad

```
5
2019-10-01 00:15:06 UTC,cart,5869134,1783999064136745198,,cosmoprofi,6.35,554342223,0b974342-1a53-41c1-a426-23130e770f4b
2019-10-01 00:17:10 UTC,cart,5787018,1487580006644188066,,,6.33,554342223,0b974342-1a53-41c1-a426-23130e770f4b
2019-10-01 00:21:02 UTC,cart,5836843,1487580009261432856,,pnb,0.71,554342223,0b974342-1a53-41c1-a426-23130e770f4b
2019-10-01 00:22:24 UTC,cart,5755171,1487580009387261981,,,2.48,554342223,0b974342-1a53-41c1-a426-23130e770f4b
2019-10-01 00:22:56 UTC,cart,5691026,1487580009387261981,,,2.86,554342223,0b974342-1a53-41c1-a426-23130e770f4b
```

Figura 2: Contenido del fichero de datos `./tests/validation/Ecommerce5.keyboard`

1. Descripción

Recordemos que todas las prácticas de este año están orientadas al problema de ventas por internet, analizando el registro de actividad de los clientes de una web de venta de productos, y elaborando informes de comportamiento de las ventas.

En las prácticas anteriores ya se implementaron las clases `DateTime`, `Event`, `EventSet`, `Pair`, `Index` así como un programa principal para integrar y usar dichas clases para realizar un estudio sobre usuarios y marcas. En Shopping3 las clases `EventSet`, e `Index` contienen arrays de objetos `Event` y `Pair` respectivamente, que se alojan y liberan de forma automática en la pila fijando el tamaño máximo a un número preestablecido de componentes `MAXEVENT`. Dependiendo de su valor, puede ser pequeño o muy grande o bien no tener suficiente memoria o bien estar desperdiciando zona de memoria de la pila. Por ello, queremos abandonar ese valor máximo por defecto, respecto a EventSet e Index. Lo que pretendemos es que la memoria ocupada por un objeto EventSet e Index se ajuste más al número de eventos de que se dispone.

1.1. Los datos

En esta práctica seguiremos trabajando con un pequeño fragmento del conjunto de datos reales (alrededor de 75,000 registros) descargados desde la plataforma Kaggle[1]. En la Figura 1 se encuentra un recordatorio del formato de dichos datos en el dataset. Recuerda que un evento puede tener algunos valores vacíos. Los únicos datos que siempre estarán presentes serán DateTime, Product ID, UserID, y SessionID.

1.2. Arquitectura de la práctica (recordatorio)

La práctica `Shopping` se ha diseñado como una arquitectura por capas, en la que las capas más internas de la misma representan las estruc-

[1] Kaggle (Abrir en navegador →)

turas más sencillas, sobre las cuales se asientan las capas más externas, con las estructuras más complejas. Durante el desarrollo de la asignatura se implementarán un total de 5 clases (2 de ellas se van a actualizar de forma significativa) que dotarán al programa de la funcionalidad deseada. Concretamente las clases que nos vamos a encontrar y que tendremos que implementar son:

A DateTime.cpp
: Implementa la clase DateTime, compuesta por año, mes, día, hora, minutos y segundos. Toda instancia de esta clase ha de ser correcta, esto es, una fecha ha de ser válida y el tiempo dentro de los rangos correctos.

B Event.cpp
: Implementa la clase Event, que contiene la información de cada acción registrada en el sitio web.

C EventSet.cpp
: Implementa la estructura para almacenar un conjunto de eventos. Como comprobaremos muchos menos que los deseables. Inicialmente usaremos arrays estáticos.

D Pair.cpp
: Estructura para almacenar una clave de búsqueda y una posición en el EventSet.

E Index.cpp
: Array de claves, que se va a utilizar como índice para la búsqueda y recuperación eficiente en el EventSet.

C, E **EventSet.cpp Index.cpp**
: Manteniendo la interfaz anterior, se cambia la implementación para alojarlas de forma más eficiente en memoria dinámica. Se revisa el diseño de cada clase para eliminar la dependencia de `MAXEVENT` y es necesario dotarlas de nuevas funcionalidades que serán transparentes al usuario de nuestra clase (usuario cliente), pues serán métodos privados y operadores familiares para el usuario cliente [2].

F Report.cpp
: Implementa la estructura para almacenar una matriz que nos va a permitir realizar estadísticas y comparativas.

Este trabajo progresivo se ha planificado en hitos sucesivos con entregas en Prado.

[2] O bien, el usuario cliente no los va a llamar explícitamente (los métodos son privados) o si los llama (ej.: `operator(=)`), no los va a percibir como operadores nuevos si no como una extensión del lenguaje.

2. Clase EventSet

La clase EventSet mantiene muchas funcionalidades previas como : `EventSet(); size(); clear() add(const Event & e); at()`... pero se ha de revisar su definición que requiere de labor de programación precisa para la gestión de memoria en el heap [3], además de otras funcionales que aparecen por su nueva representación.

2.1. Representación

Antes de comenzar a describir los distintos métodos, veamos la representación interna de la clase, y cómo queda nuestro vector de eventos. En concreto, consideraremos los siguientes atributos:

```
class EventSet { // New representation of EventSet
   Event *_events; // Set of events load correctly
   unsigned _nEvents;  // Number of available Events in the block
   unsigned _capacity; // size of the allocated block for _events
};
```

- `_events`: Es nuestro vector de Eventos en el heap, reservado en un bloque con un máximo de hasta hasta `_capacity` eventos. Por cuestiones de eficiencia, el bloque puede tener un tamaño mayor del estrictamente necesario (entendiendo este por el número de eventos efectivamente almacenadas en el mismo). Tanto es así que, si necesitamos añadir algún evento a un bloque lleno, este se debe redimensionar para dar cabida a nuevos eventos. Con el objetivo de no particionar excesivamente la memoria, cada vez que se redimensione el bloque la capacidad actual se dobla [4] y se realojan los eventos anteriores al espacio nuevo mayor.
- `_nEvents`: Número de componentes útiles en el vector, se corresponde con el `size()` del `EventSet`.
- `_capacity`: Número máximo de eventos que se podría almacenar sin necesidad de realizar nuevas reservas (realojos) de memoria.

Teniendo en cuenta todas estas restricciones, un `EventSet` debe cumplir:

1. `_capacity` es el tamaño del bloque reservado para `_events`.
2. `_events` almacena los elementos de forma consecutiva, ocupando las primeras `_nEvents` posiciones, empezando en la posición 0, por tanto en `i` hay un evento válido si `0 <= i < _nEvents`
3. `0 <= _nEvents <= _capacity`.

[3] Esta es una responsabilidad entera del programador, no hay reservas, ni liberaciones automáticas etc...

[4] Realmente se seguirá la siguiente fórmula: `2*_capacity +1` para que la primera vez que se añada un evento al menos se reserve un slot de memoria.

2.2. Especificación

A través de la especificación indicamos los métodos que permiten trabajar con un `EventSet`, el API de la clase y, el usuario cliente de la clase así como las funciones externas a esta solo podrán utilizar dichos métodos.

```
class EventSet {
    Event *_events; // Set of events load correctly
    unsigned  _nEvents; // Number of available events
    unsigned _capacity; // size of the allocated block of memory
    ... // private methods
public:
    // Constructor
    EventSet(); /** New behaviour **/

    // Consult methods
    int size() const;
    std::string to_string() const;
    const Event & at(int pos) const;
    void write(std::ofstream &os) const;
    int getCapacity() const;

    // Modifier methods
    void clear();             /** New behaviour **/
    int add(const Event & e);  /** New behaviour **/
    int add(const std::string& line);
    Event & at(int pos);
    bool read(std::ifstream &is, int nelements);

    // New methods
    /**
     * @brief Copy constructor, a new object is created from the parameter object
     * @param orig the EventSet we want a copy from
     */
    EventSet(const EventSet & orig);

    /**
     *  @brief Destructor, deallocate the allocated memory of the object
     */
    ~EventSet();

    /**
     * Assignment operator overload. This operator copies the content of other
     * EventSet into this.
     * @param one EventSet to be copied
     * @return Reference to this, in order to chain the operator
     */
    EventSet & operator=(const EventSet & one);

};
```

Como siempre, puedes definir todos aquellos métodos privados que estimes oportunos. En particular, para evitar duplicar código a la hora de reservar y liberar memoria, se recomienda implementar los métodos privados tipo (`allocate()`, `clean()` `copy(...)`, `reallocate()`).

Para ilustrar la progresión de ocupación de memoria, veamos el siguiente ejemplo para ubicar 14 Eventos en el EventSet, veamos las siguientes correspondencias entre capacidad y size:

```
capacity 0 ... updating capacity
capacity 1 size 1  ... updating capacity
capacity 3 size 2   size 3  ... updating capacity
capacity 7 size 4   size 5   size 6   size 7 ... updating capacity
capacity 15 size 8   size 9  size 10  size 11 size 12 size 13  size 14
```

3. Clase Index

3.1. Representación

De la misma forma que se hiciera con `EventSet`, la declaración de la clase `Index` cambia para incorporar una capacidad de memoria, y el vector se limita a un puntero a `Pair`.

```
class Index {
private:
    Pair *_entries;     // array of Pair
    unsigned _nEntries; // Number of active entries
    unsigned _capacity; // Number of current allocated positions in memory
    int _onBrand; //  1 on Brand 0 on UserID (bool for 2 indexes o int for more indexes)
```

3.2. Especificación

De la misma forma, la especificación que se muestra a continuación contiene el conjunto de métodos para esta revisión de la clase `Index` manteniéndose idénticos los de consulta, actualizando su comportamiento muchos de los modificadores y apareciendo nuevos métodos para el buen comportamiento del TDA[5] con memoria dinámica.

```
class Index {
private:
    Pair *_entries; // array of Pair
    unsigned _nEntries;  // Number of active entries
    unsigned _capacity; // Number of current allocated positions in memory
    int _onBrand; //  1 on Brand 0 on UserID (bool for 2 indexes o int for more indexes)
    ... // metodos privados
public:
    // Constructor
    // UsersIndex onBrand=0  o BrandType onBrand=1, defaults to 0
    Index(int onBrand=0);   /** New behaviour **/

    // Consult methods
    int size() const;
    inline bool isEmpty() {  return size()==0; }
    int getIOnWhich() const;
    const Pair & at(int pos) const;
    int lower_bound(const std::string & key) const;
    int upper_bound(const std::string & key) const;
    void print()const;
    std::string to_string() const;
    std::string reportData() const;
    void write(std::ofstream &os) const;

    // Modifier methods
    void build(const EventSet & evSet, int onBrand); /** New behaviour **/
    void clear();   /** New behaviour **/
    Pair & at(int pos);
    int add(const Pair & pair); /** New behaviour **/

    // New methods
    /**
     * @brief Queries the _capacity of the object, that is, the current size of the block
     * memory for the  eventSet.
     * @return  int capacity, the current reserved space for the object.
     */
    int getCapacity() const;   /** New behaviour **/

    /**
     * @brief Copy constructor, a new object is created from the parameter object
     * @param orig the Index we want a copy from
     */
    Index(const Index & orig);

    /**
     * Assignment operator overload. This operator copies the content of one Index into this.
     * @param one Index to be copied
     * @return Reference to this, in order to chain the operator
     */
    Index &operator=(const Index &one);

    /**
     *  @brief Destructor, deallocate the allocated memory for the object
     */
    ~Index();
};
```

4. Shopping4, práctica a entregar

El programa principal es idéntico al de la práctica Shopping3, mostrándose el estudio de compras por días, usuarios y marcas obtenidos a partir de un histórico de compras así como unas consultas a los datos según

[5] TDA significa Tipo de Dato Abstracto

los parámetros del main. El conjunto de tests será el mismo que en el caso de Shopping 3 con la salvedad de que el test `03_Advanced` relativo a `EventSet_add_event_full` deberá comentarse, ya que el número máximo de eventos ya no estará limitado al límite impuesto por `MAXEVENT`.

No obstante, no se deberá eliminar dicha constante ya que en el programa principal (main) hay aún definición de vectores que dependen de dicho límite como los obtenidos por el uso de la función `findUnique()` o bien los vectores que contienen conteos, etc. En estos casos se haría necesario el uso de estructuras dinámicas para vectores de objetos tipo `int, string,` etc. Para ello, el/la estudiante con inquietud puede intentar modificar su programa principal para incluir este manejo dinámico, definiendo sus propias clases dinámicas para los tipos de datos que necesite, o bien utilizar la clase `vector` proporcionada por C++. Se puede consultar información sobre esta clase en los siguientes recursos:

- https://www.cplusplus.com/reference/vector/vector/
- https://aprende.olimpiada-informatica.org/cpp-vector
- https://riptutorial.com/cplusplus/topic/511/std--vector

Se recomienda el uso de la herramienta **valgrind** para ver si hay algún tipo de error/fuga de memoria. Para ello puede consultarse el guión sobre valgrind puesto a disposición en Prado.

4.1. Entrega de la práctica

Una vez terminada la práctica que, al menos, haya superado los tests básicos, se debe hacer un zip (se sugiere utilizar la script `runZipProject.sh`) excluyendo las carpetas `./dist/`, `./build/`, `./nbproject/private/`, `./doc/html/` y `./dos/latex/` y subirla a Prado antes de la fecha de cierre de la entrega.

4.2. Tests completos de la práctica

Esta cuarta práctica mantiene los tres niveles de testeo, que incluyen tanto tests unitarios como de integración y que incluyen los tests de la práctica anterior quitando el test avanzado `EventSet_add_event_full` quedando por tanto la estructura de tests como sigue:

- Nivel Básico: 27 tests
- Nivel Intermedio: 10 tests
- Nivel Avanzado: 21 tests. Varios de ellos hacen un chequeo de memoria adicional, para comprobar que ningún vector de datos se sale más allá de sus límites.

Y este debería ser el resultado del test de la aplicación satisfaciendo todos los tests que se han diseñado (en azul aparece la llamada a los tests desde una terminal del proyecto).

make test

```
[==========] Running 59 tests from 3 test suites.
[----------] Global test environment set-up.
[----------] 27 tests from _01_Basics
[ RUN      ] _01_Basics.DateTime_Constructors
[       OK ] _01_Basics.DateTime_Constructors (5 ms)
[ RUN      ] _01_Basics.DateTime_getters
[       OK ] _01_Basics.DateTime_getters (6 ms)
[ RUN      ] _01_Basics.DateTime_set
[       OK ] _01_Basics.DateTime_set (2 ms)
[ RUN      ] _01_Basics.DateTime_sameDay
[       OK ] _01_Basics.DateTime_sameDay (5 ms)
[ RUN      ] _01_Basics.Event_ConstructorBase
[       OK ] _01_Basics.Event_ConstructorBase (4 ms)
[ RUN      ] _01_Basics.Event_Setters_getters
[       OK ] _01_Basics.Event_Setters_getters (9 ms)
[ RUN      ] _01_Basics.EventSet_Constructor
[       OK ] _01_Basics.EventSet_Constructor (3 ms)
[ RUN      ] _01_Basics.EventSet_add_event
[       OK ] _01_Basics.EventSet_add_Event (5 ms)
[ RUN      ] _01_Basics.EventSet_add_line
[       OK ] _01_Basics.EventSet_add_line (4 ms)
[ RUN      ] _01_Basics.EventSet_at_basic
[       OK ] _01_Basics.EventSet_at_basic (4 ms)
[ RUN      ] _01_Basics.Pair_Constructors
[       OK ] _01_Basics.Pair_Constructors (4 ms)
[ RUN      ] _01_Basics.Pair_isEmpty
[       OK ] _01_Basics.Pair_isEmpty (2 ms)
[ RUN      ] _01_Basics.Pair_setters
[       OK ] _01_Basics.Pair_setters (1 ms)
[ RUN      ] _01_Basics.Pair_getters
[       OK ] _01_Basics.Pair_getters (0 ms)
[ RUN      ] _01_Basics.Index_Constructors
[       OK ] _01_Basics.Index_Constructors (1 ms)
[ RUN      ] _01_Basics.Index_getIOnWhich
[       OK ] _01_Basics.Index_getIOnWhich (1 ms)
[ RUN      ] _01_Basics.Index_clear
[       OK ] _01_Basics.Index_clear (2 ms)
[ RUN      ] _01_Basics.Integrated_5_records
[ MEMCHECK ] ECommerce5-valgrind
[       OK ] ECommerce5-valgrind
[       OK ] _01_Basics.Integrated_5_records (28 ms)
[ RUN      ] _01_Basics.Integrated_30_records
[ MEMCHECK ] ECommerce30-valgrind
[       OK ] ECommerce30-valgrind
[       OK ] _01_Basics.Integrated_30_records (24 ms)
[ RUN      ] _01_Basics.Integrated_41_records
[ MEMCHECK ] ECommerce41-valgrind
[       OK ] ECommerce41-valgrind
[       OK ] _01_Basics.Integrated_41_records (31 ms)
[ RUN      ] _01_Basics.Integrated_162_records
[ MEMCHECK ] ECommerce162-valgrind
[       OK ] ECommerce162-valgrind
[       OK ] _01_Basics.Integrated_162_records (37 ms)
[ RUN      ] _01_Basics.Integrated_926_records
[ MEMCHECK ] ECommerce926-valgrind
[       OK ] ECommerce926-valgrind
[       OK ] _01_Basics.Integrated_926_records (139 ms)
[ RUN      ] _01_Basics.Integrated_Args_5_records
[ MEMCHECK ] ECommerce5-valgrind
[       OK ] ECommerce5-valgrind
[       OK ] _01_Basics.Integrated_Args_5_records (27 ms)
[ RUN      ] _01_Basics.Integrated_Args_30_records
[ MEMCHECK ] ECommerce30-valgrind
[       OK ] ECommerce30-valgrind
[       OK ] _01_Basics.Integrated_Args_30_records (28 ms)
[ RUN      ] _01_Basics.Integrated_Args_41_records
[ MEMCHECK ] ECommerce41-valgrind
[       OK ] ECommerce41-valgrind
[       OK ] _01_Basics.Integrated_Args_41_records (28 ms)
[ RUN      ] _01_Basics.Integrated_Args_162_records
[ MEMCHECK ] ECommerce162-valgrind
[       OK ] ECommerce162-valgrind
[       OK ] _01_Basics.Integrated_Args_162_records (29 ms)
[ RUN      ] _01_Basics.Integrated_Args_926_records
[ MEMCHECK ] ECommerce926-valgrind
[       OK ] ECommerce926-valgrind
[       OK ] _01_Basics.Integrated_Args_926_records (134 ms)
[----------] 27 tests from _01_Basics (563 ms total)

[----------] 10 tests from _02_Intermediate
[ RUN      ] _02_Intermediate.DateTime_isBefore
[       OK ] _02_Intermediate.DateTime_isBefore (4 ms)
[ RUN      ] _02_Intermediate.DateTime_weekDay
[       OK ] _02_Intermediate.DateTime_weekDay (4 ms)
[ RUN      ] _02_Intermediate.Event_getField
[       OK ] _02_Intermediate.Event_getField (4 ms)
[ RUN      ] _02_Intermediate.EventSet_add_event_partial
[       OK ] _02_Intermediate.EventSet_add_event_partial (4 ms)
[ RUN      ] _02_Intermediate.EventSet_at_intermediate
[       OK ] _02_Intermediate.EventSet_at_intermediate (5 ms)
[ RUN      ] _02_Intermediate.Index_3x3_just_build
[       OK ] _02_Intermediate.Index_3x3_just_build (4 ms)
[ RUN      ] _02_Intermediate.Index_B_BxU_build_at
[       OK ] _02_Intermediate.Index_B_BxU_build_at (7 ms)
[ RUN      ] _02_Intermediate.Index_U_BxU_build_at
[       OK ] _02_Intermediate.Index_U_BxU_build_at (4 ms)
[ RUN      ] _02_Intermediate.Integrated_EMPTY
```

```
[ MEMCHECK ] EMPTY-valgrind
[       OK ] EMPTY-valgrind
[       OK ] _02_Intermediate.Integrated_EMPTY (44 ms)
[ RUN      ] _02_Intermediate.Integrated_ErrorLoading
[ MEMCHECK ] ErrorLoading-valgrind
[       OK ] ErrorLoading-valgrind
[       OK ] _02_Intermediate.Integrated_ErrorLoading (40 ms)
[----------] 10 tests from _02_Intermediate (121 ms total)

[----------] 21 tests from _03_Advanced
[ RUN      ] _03_Advanced.DateTime_BadValues
[       OK ] _03_Advanced.DateTime_BadValues (14 ms)
[ RUN      ] _03_Advanced.Event_setType_Bad_Values
[       OK ] _03_Advanced.Event_setType_Bad_Values (5 ms)
[ RUN      ] _03_Advanced.Event_Others_Bad_Values
[       OK ] _03_Advanced.Event_Others_Bad_Values (6 ms)
[ RUN      ] _03_Advanced.EventSet_at_advanced
[       OK ] _03_Advanced.EventSet_at_advanced (7 ms)
[ RUN      ] _03_Advanced.EventSet_externalfunctions
[       OK ] _03_Advanced.EventSet_externalfunctions (6 ms)
[ RUN      ] _03_Advanced.EventSet_write
[       OK ] _03_Advanced.EventSet_write (4 ms)
[ RUN      ] _03_Advanced.EventSet_read
[       OK ] _03_Advanced.EventSet_read (4 ms)
[ RUN      ] _03_Advanced.Index_U_BxU_bounds
[       OK ] _03_Advanced.Index_U_BxU_bounds (4 ms)
[ RUN      ] _03_Advanced.Index_B_BxU_bounds
[       OK ] _03_Advanced.Index_B_BxU_bounds (5 ms)
[ RUN      ] _03_Advanced.Index_add
[       OK ] _03_Advanced.Index_add (6 ms)
[ RUN      ] _03_Advanced.Index_B_BxU_rawFilterIndex
[       OK ] _03_Advanced.Index_B_BxU_rawFilterIndex (3 ms)
[ RUN      ] _03_Advanced.Index_U_BxU_rawFilterIndex
[       OK ] _03_Advanced.Index_U_BxU_rawFilterIndex (4 ms)
[ RUN      ] _03_Advanced.Index_Type_BxU_rawFilterIndex
[       OK ] _03_Advanced.Index_Type_BxU_rawFilterIndex (5 ms)
[ RUN      ] _03_Advanced.Index_DateTime_BxU_rawFilterIndex
[       OK ] _03_Advanced.Index_DateTime_BxU_rawFilterIndex (2 ms)
[ RUN      ] _03_Advanced.Index_BxU_sumPrice
[       OK ] _03_Advanced.Index_BxU_sumPrice (8 ms)
[ RUN      ] _03_Advanced.Integrated_ErrorData
[ MEMCHECK ] ErrorData-valgrind
[       OK ] ErrorData-valgrind
[       OK ] _03_Advanced.Integrated_ErrorData (627 ms)
[ RUN      ] _03_Advanced.Integrated_ErrorSaving
[ MEMCHECK ] ErrorSaving-valgrind
[       OK ] ErrorSaving-valgrind
[       OK ] _03_Advanced.Integrated_ErrorSaving (646 ms)
[ RUN      ] _03_Advanced.Integrated_Args_no_open
[ MEMCHECK ] ECommerce926-valgrind
[       OK ] ECommerce926-valgrind
[       OK ] _03_Advanced.Integrated_Args_no_open (639 ms)
[ RUN      ] _03_Advanced.Integrated_Args_error_data
[ MEMCHECK ] ECommerce162-valgrind
[       OK ] ECommerce162-valgrind
[       OK ] _03_Advanced.Integrated_Args_error_data (668 ms)
[ RUN      ] _03_Advanced.Integrated_Args_error_missing_arg
[ MEMCHECK ] ECommerce162-valgrind
[       OK ] ECommerce162-valgrind
[ MEMCHECK ] ECommerce162-valgrind
[       OK ] ECommerce162-valgrind
[       OK ] _03_Advanced.Integrated_Args_error_missing_arg (1202 ms)
[ RUN      ] _03_Advanced.Integrated_Args_error_bad_arg
[ MEMCHECK ] ECommerce162-valgrind
[       OK ] ECommerce162-valgrind
[       OK ] _03_Advanced.Integrated_Args_error_bad_arg (606 ms)
[----------] 21 tests from _03_Advanced (4471 ms total)

[----------] Global test environment tear-down
[==========] 58 tests from 3 test suites ran. (18038 ms total)
[  PASSED  ] 58 tests.
```