Creación de pruebas para una clase de C++

En este documento describo la programación de un conjunto de pruebas para una clase llamada Calendario. Se utiliza el framework de pruebas CPPUNIT para construir las pruebas.

La clase Calendario

El archivo Calendario.h se muestra a continuación:

```
//Calendario.h
#ifndef CALENDARIO_H
#define CALENDARIO_H

typedef
struct SetDIntType{
  int n;
  int *intPt;
  SetDIntType():n(0),intPt(NULL){ }
  bool operator==(SetDIntType &sdit)const{
    bool tmp0=true;
    bool tmp1=(n==sdit.n);
    for(int i=0;i<n;i++){
      if(*(intPt+i)!=*(sdit.intPt+i)){
        tmp0=false;
        break;
      }
    }
    return (tmp0 && tmp1);
  }
}SetDIntType;


class Calendario{
public:
  void mostrar_fechas(string dia,string mes);
  Calendario(int year):numdanio(year){ }
  int get_anio(){
    return numdanio;
  }
  //...
  /**
   Obtiene los numeros de dia de las fechas de los
   dias d en el mes m.
   */
  SetDIntType* obtener_nums_ddia(string d,string m);
```

```
  void print_SetDIntYFecha(string d,string m,SetDIntType* SDI);

  /**
   Obtiene el indice correspondiente al mes month.
   Si month="enero", debe devolver 0,
   Si month="febrero", debe devolver 1,
   Si month="marzo", debe devolver 2,
   etc.
   */
  int index_delmes(string month);

  /**
   Devuelve el numero de dia de la primera fecha
   del mes con indice i (devuelto por
   index_delmes(string month)), en la que el dia
   es el string day.
   */
  int primera_fecha_delmes(string day,int i);

  /**
    Cumplimenta el SetDIntType al que apunta el apuntador r.
    Esto es, despues de llamar a este metodo, r->n contendra
    la cantidad de veces que se presenta el dia de la semana
    correspondiente a la fecha j (j es un int que corresponde
    al primer lunes, martes, miercoles etc.
    j\in{1,2,3,4,5,6,7}), en el mes con indice i. Mientras que
    el apuntador r->intPt apuntara a los numeros de dia en las
    fechas tales que el dia coincide con el dia de la fecha j
    del mes con indice i.
    @param i: index del mes; 0 enero, 1 febrero, 2 marzo, etc.
    @param j: primera fecha del mes correspondiente al dia de
              la semana cuyas fechas que estamos buscando.
    @param r: Conjunto de enteros.
 */
  void fill_SDIT(int i,int j,SetDIntType* r);

  int numdanio;  //numero de anio
};//end class Calendario
#endif
```

La implementación de esta clase se tendrá en el archivo `Calendario.cpp`. Este último es el que se pondrá a prueba con las pruebas descritas en este documento. En este caso, para construir las pruebas se usarán tres archivos: `hacer_pruebas.cpp` y el par `prueba.h`, `prueba.cpp`.
El archivo `hacer_pruebas.cpp` es:

```
//hacer_pruebas.cpp
```

```
//#include <cppunit/CompilerOutputter.h>
//#include <cppunit/extensions/TestFactoryRegistry.h>
#include <iostream>
using std::string;
#include <cppunit/TestSuite.h>
#include <cppunit/ui/text/TestRunner.h>

#include "Calendario.h"
#include "prueba.h"

using namespace std;

int main(){
  //Agregar esta suite de prueba a la lista de pruebas a ejecutarse.
  CppUnit::TextUi::TestRunner runner;
  cout<<"Creando suite de pruebas"<<endl;
  runner.addTest(TestCalendario::suite());

  //Correr pruebas
  cout<<"Corriendo pruebas de unidad..."<<endl;
  bool wasSuccessfull=runner.run();

  //Devolver codigo de error
  if(wasSuccessfull)
    return 0;
  else
    return 1;
}//end main()
```

En el archivo anterior se crea una suite de pruebas a través de la ejecución del método `static` (o método de clase) `TestCalendario::suite()`. La clase `TestCalendario` hereda de la clase `CppUnit::TestFixture`, la declaración de la clase `TestCalendario` está en el archivo `prueba.h` que incluyo a continuación:

```
//prueba.h
#ifndef TEST_CALENDARIO
#define TEST_CALENDARIO
//#include <cppunit/extensions/HelperMacros.h>
#include <iostream>
#include <cppunit/TestFixture.h>
#include <cppunit/TestAssert.h>
#include <cppunit/TestCaller.h>
#include <cppunit/TestSuite.h>
#include <cppunit/TestCase.h>
#include <fstream>

#include "Calendario.h"
```

```
using namespace std;

class TestCalendario : public CppUnit::TestFixture {
private:
  Calendario* testCalendario;
public:
  TestCalendario():testCalendario(NULL){}
  virtual ~TestCalendario(){
    delete testCalendario;
  }
  static CppUnit::Test *suite(); /*en este metodo se agregan las pruebas*/

  void setUp() {}//Setup method
  void tearDown() {}//Teardown method
protected: /*estas son las pruebas*/
  void testCreacionDeUnCalendario();
  void test_Calendario_index_delmes();
  void test_primera_fecha_delmes();
  void test_fill_SDIT();
};//end class TestCalendario

#endif /*TEST_CALENDARIO*/
```

La clase `TestCalendario` está implementada en el archivo `prueba.cpp`. En `prueba.cpp` se pone a prueba los métodos `index_delmes(string)`, `primera_fecha_delmes(string)`, y el método `fill_SDIT(int,int,SetDIntType*)` de la clase `Calendario`. El archivo `prueba.cpp` es

```
//prueba.cpp
#include <iostream>
using std::string;
//#include <cppunit/TestFixture.h>
//#include <cppunit/TestAssert.h>
//#include <cppunit/TestCaller.h>
#include <cppunit/TestSuite.h>  /*CppUnit*/
#include <stdlib.h>/*malloc()*/
#include "prueba.h"
extern string ARREGLO[][7];
extern int TamDMes[];

CppUnit::Test* TestCalendario::suite(){
  CppUnit::TestSuite *suiteOfTests=new CppUnit::TestSuite("Prueba de Calendario");

  suiteOfTests->addTest(new CppUnit::TestCaller<TestCalendario>
    ("Test 1 - Crear Calendario 2016.", &TestCalendario::testCreacionDeUnCalendario));
```

```
  suiteOfTests->addTest(new CppUnit::TestCaller<TestCalendario>
    ("Test 2 - Determinar si el metodo Calendario::index_delmes(string) \
funciona bien.",&TestCalendario::test_Calendario_index_delmes));

  suiteOfTests->addTest(new CppUnit::TestCaller<TestCalendario>("Test 3 - Determinar \
si el metodo Calendario::primera_fecha_delmes(string,int) funciona bien.",
&TestCalendario::test_primera_fecha_delmes));

  suiteOfTests->addTest(new CppUnit::TestCaller<TestCalendario>("Test 4 - Determinar \
si el metodo Calendario::fill_SDIT(int i,int j,SetDIntType* r) funciona bien.",
&TestCalendario::test_fill_SDIT));

  /* Aqui se pueden agregar mas tests */
  return suiteOfTests;
}


void TestCalendario::testCreacionDeUnCalendario(){
  int year=2016;
  Calendario Cal2016(year);
  //CPPUNIT_ASSERT_EQUAL(CAS.get_delegacion(), delegacion);
  CPPUNIT_ASSERT_EQUAL(Cal2016.get_anio(), year);
}


void TestCalendario::test_Calendario_index_delmes(){
  Calendario *calPt;
  calPt=new Calendario(2016);
  /*Lo que se espera, lo que se obtiene*/
  CPPUNIT_ASSERT_EQUAL(0,calPt->index_delmes("enero"));
  CPPUNIT_ASSERT_EQUAL(1,calPt->index_delmes("febrero"));
  CPPUNIT_ASSERT_EQUAL(2,calPt->index_delmes("marzo"));
  CPPUNIT_ASSERT_EQUAL(3,calPt->index_delmes("abril"));
  CPPUNIT_ASSERT_EQUAL(4,calPt->index_delmes("mayo"));
  CPPUNIT_ASSERT_EQUAL(5,calPt->index_delmes("junio"));
  CPPUNIT_ASSERT_EQUAL(6,calPt->index_delmes("julio"));
  CPPUNIT_ASSERT_EQUAL(7,calPt->index_delmes("agosto"));
  CPPUNIT_ASSERT_EQUAL(8,calPt->index_delmes("septiembre"));
  CPPUNIT_ASSERT_EQUAL(9,calPt->index_delmes("octubre"));
  CPPUNIT_ASSERT_EQUAL(10,calPt->index_delmes("noviembre"));
  CPPUNIT_ASSERT_EQUAL(11,calPt->index_delmes("diciembre"));
}

int primera_fecha_delmes(string day,int i){
  if(ARREGLO[i][0]==day) return 7;
  if(ARREGLO[i][1]==day) return 1;
```

```
  if(ARREGLO[i][2]==day) return 2;
  if(ARREGLO[i][3]==day) return 3;
  if(ARREGLO[i][4]==day) return 4;
  if(ARREGLO[i][5]==day) return 5;
  if(ARREGLO[i][6]==day) return 6;
  return -1;/*nunca deberia llegarse aqui*/
}
string DIA[]={"Domingo","Lunes","Martes","Miercoles","Jueves","Viernes","Sabado"};

void TestCalendario::test_primera_fecha_delmes(){
  int i,j;
  Calendario *calPt=new Calendario(2016);
  //Ahora probamos 7*12 combinaciones posibles de dias de la semana y mes.
  //P.ej. ("Domingo",0),("Domingo",1),...,("Lunes",0),("Lunes",1),..., etc.
  for(i=0;i<7;i++){
    for(j=0;j<12;j++){
      CPPUNIT_ASSERT_EQUAL(primera_fecha_delmes(DIA[i],j),
                           calPt->primera_fecha_delmes(DIA[i],j));
    }
  }
}

void fill_SDIT(int i,int j,SetDIntType* r){
  int cnt=1,k=j;
  while((k=k+7)<=TamDMes[i])cnt++;
  r->n=cnt;
  k=j;
  r->intPt=(int*)malloc((r->n)*sizeof(int));
  for(int m=0;m<r->n;m++){
    *(r->intPt+m)=k;
    k+=7;
  }
}

void TestCalendario::test_fill_SDIT(){
  Calendario *calPt=new Calendario(2016);
//  SetDIntType R,R1;
  SetDIntType* r=(SetDIntType*)malloc(sizeof(SetDIntType));
  SetDIntType* r1=(SetDIntType*)malloc(sizeof(SetDIntType));
//  SetDIntType* r=&R;
//  SetDIntType* r1=&R1;
  int i,j1,j2,k;
  string d;
  for(i=0;i<12;i++){
    for(k=0;k<7;k++){
      d=DIA[k];
```

```
      j1=primera_fecha_delmes(d,i);j2=calPt->primera_fecha_delmes(d,i);
      fill_SDIT(i,j1,r); calPt->fill_SDIT(i,j2,r1);
//      CPPUNIT_ASSERT(R==R1);
      CPPUNIT_ASSERT(*r==*r1);
    }
  }

//  j=primera_fecha_delmes("Domingo",0);
//  fill_SDIT(0,j,&R);calPt->fill_SDIT(0,j,&R1);
//  CPPUNIT_ASSERT(R==R1);
}
```

El objetivo de estas pruebas de unidad es poder probar diferentes imlementaciones de la clase Calendario. Esto puede servir por ejemplo, para revisar implementaciones de los métodos de la clase Calendario que se utilicen como evidencia en un curso en donde se evalue el aprendizaje de la programación en lenguaje C++. Una posible implementación de la clase `Calendario` está dada en el siguiente archivo:

```
//Calendario.cpp
#include <iostream>
#include <stdlib.h>/*malloc()*/
using std::string;
using std::cout;
using std::endl;
#include "Calendario.h"

string ARREGLO[][7]={
  {"Jueves","Viernes","Sabado","Domingo","Lunes","Martes","Miercoles"}, /*enero*/
  {"Domingo","Lunes","Martes","Miercoles","Jueves","Viernes","Sabado"}, /*febrero*/
  {"Lunes","Martes","Miercoles","Jueves","Viernes","Sabado","Domingo"}, /*marzo*/
  {"Jueves","Viernes","Sabado","Domingo","Lunes","Martes","Miercoles"}, /*abril*/
  {"Sabado","Domingo","Lunes","Martes","Miercoles","Jueves","Viernes"}, /*mayo*/
  {"Martes","Miercoles","Jueves","Viernes","Sabado","Domingo","Lunes"}, /*junio*/
  {"Jueves","Viernes","Sabado","Domingo","Lunes","Martes","Miercoles"}, /*julio*/
  {"Domingo","Lunes","Martes","Miercoles","Jueves","Viernes","Sabado"}, /*agosto*/
  {"Miercoles","Jueves","Viernes","Sabado","Domingo","Lunes","Martes"}, /*septiembre*/
  {"Viernes","Sabado","Domingo","Lunes","Martes","Miercoles","Jueves"}, /*octubre*/
  {"Lunes","Martes","Miercoles","Jueves","Viernes","Sabado","Domingo"}, /*noviembre*/
  {"Miercoles","Jueves","Viernes","Sabado","Domingo","Lunes","Martes"}  /*diciembre*/
};

/*Cantidades de dias de los meses del anio (valido para anios bisiestos)*/
int TamDMes[]={
  31,/*enero*/
  29,/*febrero*/
  31,/*marzo*/
```

```
  30,/*abril*/
  31,/*mayo*/
  30,/*junio*/
  31,/*julio*/
  31,/*agosto*/
  30,/*septiembre*/
  31,/*octubre*/
  30,/*noviembre*/
  31 /*diciembre*/
};

void Calendario::mostrar_fechas(string d,string m){
  SetDIntType* sdi=obtener_nums_ddia(d,m);
  print_SetDIntYFecha(d,m,sdi);
}

void print_SetDIntYFecha(string d,string m,SetDIntType* SDI){
  for(int i=0;i<SDI->n;i++){
    cout<<SDI->intPt[i]<<" de "<<m<<" de 2016"<<endl;
  }
}

SetDIntType* Calendario::obtener_nums_ddia(string d,string m){
  SetDIntType* r=(SetDIntType*)malloc(sizeof(SetDIntType));
  int i,j;  /* i: index del mes en el arreglo ARREGLO*/
  i=index_delmes(m);
  if(i>=0 && i<12){
    j=primera_fecha_delmes(d,i);
  }
  fill_SDIT(i,j,r);
  return r;
}

string MES[]={"enero","febrero","marzo","abril","mayo","junio","julio","agosto",
              "septiembre","octubre","noviembre","diciembre"};

int Calendario::index_delmes(string month){
  for(int i=0;i<12;i++){
    if(month==MES[i])
      return i;
  }
  return -1;/*no se encontro la cadena*/
}

void Calendario::print_SetDIntYFecha(string d,string m,SetDIntType* SDI){
  for(int i=0;i<SDI->n;i++){
```

```
    cout<<SDI->intPt[i]<<" de "<<m<<" de 2016"<<endl;
  }
}

int Calendario::primera_fecha_delmes(string day,int i){
  if(ARREGLO[i][0]==day) return 7;
  if(ARREGLO[i][1]==day) return 1;
  if(ARREGLO[i][2]==day) return 2;
  if(ARREGLO[i][3]==day) return 3;
  if(ARREGLO[i][4]==day) return 4;
  if(ARREGLO[i][5]==day) return 5;
  if(ARREGLO[i][6]==day) return 6;
  return -1;/*nunca deberia llegarse aqui*/
}

void Calendario::fill_SDIT(int i,int j,SetDIntType* r){
  int cnt=1,k=j;
  while((k=k+7)<=TamDMes[i])cnt++;
  r->n=cnt;
  k=j;
  r->intPt=(int*)malloc((r->n)*sizeof(int));
  for(int m=0;m<r->n;m++){
    *(r->intPt+m)=k;
    k+=7;
  }
}
```

En este párrafo describo para qué se usa cada uno de los archivos descritos en el presente documento.

Calendario.h Especifica la clase cuya implementación se pondrá a prueba.

Calendario.cpp Contiene la implementación de la clase que se pone a prueba.

prueba.h Declara la clase TestCalendario, la cual hereda de la clase `CppUnit::TestFixture`. En este ejemplo, se colocaron en la declaración de la clase `TestCalendario` solo los prototipos de las pruebas

- `void testCreacionDeUnCalendario();`
- `void test_Calendario_index_delmes();`
- `void test_primera_fecha_delmes();`
- `void test_fill_SDIT();`

prueba.cpp En este archivo se agregan las pruebas a la suite de pruebas que devuelve el método de clase `static CppUnit::Test *suite();`. Nótese que cada prueba se agrega individualmente con una sentencia como esta

```
//...
  suiteOfTests->addTest(new CppUnit::TestCaller<TestCalendario>
    ("Test 1 - Crear Calendario 2016.",
     &TestCalendario::testCreacionDeUnCalendario));
//...
```

Y se programan las cuatro pruebas.

hacer_pruebas.cpp En este archivo se tiene la función `main` con la que se corren las pruebas programadas en el binomio `prueba.h/prueba.cpp`.