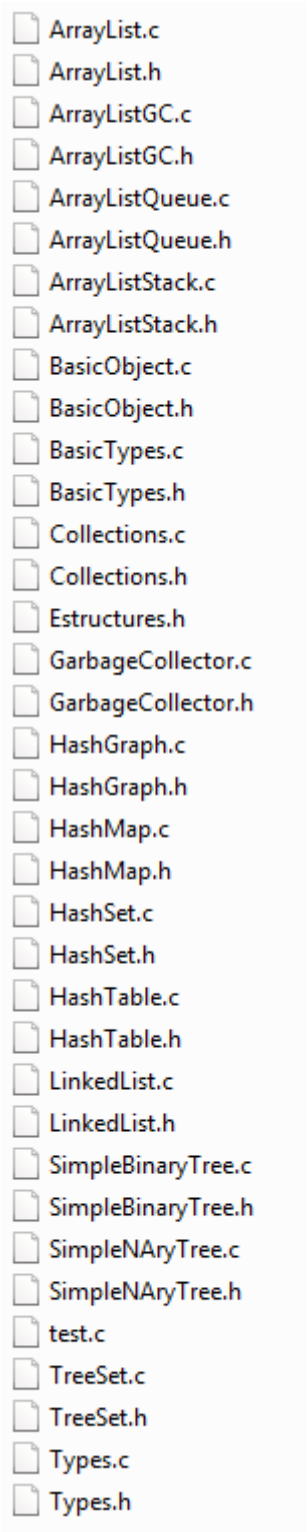
XTADs.v1: Librería de Tipos Abstractos de Datos para el Lenguaje C

F. de la Rosa Troyano



Listado de código fuente**.**

- ArrayList.c y ArrayList.h: Código de una Lista implementada mediante arrays.

- ArrayListGC.c y ArrayListGC.h: Código de la Lista utilizada por el recolector de basura..

- ArrayListStack.c y ArrayListStack.h: Código de una Pila implementada con una Lista.

- ArrayListQueue.c y ArrayListQueue.h: Código de una Cola implementada con una Lista.

- BasicObject.c y BasicObject.h: Definición del objeto genérico BasicObject. Todos los TAD son del tipo BasicObject.

- BasicTypes.c y BasicTypes.h: Definición de algunos tipos útiles para guardar dentro de los TAD. Define los tipos Int y String.

- Collection.c y Collection.h: Define los TAD relacionados con las colecciones: List, Set, Collection y Map.

- Estructures.c: Define estructuras de TAD más complejas Graph, BinaryTree, NAryTree y Table.

- GarbageCollector.c y GarbageCollector.h: Implementa el recolector de basura.

- HashGraph.c y HashGraph.h: Implementación un grafo con mapas.

- HashMap.c y HashMap.h: Implementación un map.

- HashSet.c y HashSet.h: Implementación un conjunto con un mapa.

- HashTable.c y HashTable.h: Implementación una tabla con un mapa.

- LinkedList.c y LinkedList.h: Implementación un lista enlazada.

- SimpleBinaryTree.c y SimpleBinaryTree.h: Implementación un árbol binario.

- SimpleNAryTree.c y SimpleNaryTree.h: Implementación un árbol n-ario.

- TreeSet.c y TreeSet.h: Implementación un árbol ordenado.

- Types.c y Types.h: Implementación de excepciones.

- test.c: Implementación test de pruebas..

Ejemplos de uso de las librerías

En esta sección se muestran ejemplos de uso de los distintos Tipos Abstractos de Datos implementados en XTADs.

Ejemplo: Creación y uso de Cadenas o Strings

```
void testString() {


    puts("***************************");
    puts("***** TEST STRING ****");
    puts("***************************");
    _gc_println();
    _gc_new();
    _gc_println();

    String sss1 = newString("hola");
    String sss2 = newString("hola3");

    puts(_toString(sss1));
    puts(_toString(sss2));


    _set(sss1,"totala");
    pchar saux = _toString(sss2);
    sss1 = _concat(sss1,saux);
    sss1 = _concat(sss1,"pppooopp");

    puts(_toString(sss1));

    printf("%d\n",_length(sss1));
    puts(_toString(sss2));

    printf("%d\n",_indexOf(sss1,"hola3"));
    printf("HC: %d\n",_hashcode(sss1));

    _gc_println();

    _gc_flush();
    _gc_println();
    _gc_destroyLast();
    _gc_println();
}
```

Ejemplo: Creación de una ArrayList donde se almacenan tipos Int

```
void testArrayListInt() {
        puts("****************************************");
           puts("***** TEST LISTA ARRAY DE ENTEROS  ****");
           puts("****************************************");
   // _gc_new();
       List l = newArrayList();

           printf("size: %d\n", _size(l));
              _add(l,newInt(1));
              _add(l,newInt(2));
              _add(l,newInt(3));
              _add(l,newInt(2));
              _add(l,newInt(1));
              _add(l,newInt(0));
              _add(l,newInt(-1));


              Iterator it = _iterator(l);
              while(_hasNext(it)) {
                     BasicObject ob = _next(it);
                     puts(_toString(ob));
                     _println(ob);
              }

           printf("size:%d\n", _size(l));

           puts(_toString(l));
           _println(l);

           BasicObject ba = _removeIndex(l,2);
           _destroy(ba);
           ba = _removeIndex(l,2);
           _destroy(ba);

           printf("size:%d\n", _size(l));
           puts(_toString(l));
           _println(l);

     //    _gc_destroyLast();

           fflush(stdout);

}
```

Ejemplo: Creación y Uso de Listas

```
void testList(List (*createList)(), const pchar ntest) {
    puts("***************************");
    printf("***** %s ****\n",ntest);
    puts("***************************");

    _gc_new();

    List l = createList();

    printf("%d\n", _size(l));

    _add(l,newString("hola1"));
    _add(l,newString("hola2"));
    _add(l,newString("hola3"));
    _add(l,newString("hola4"));
    _add(l,newString("hola4"));
    _add(l,newString("hola5"));
    puts(_toString(l));

    Iterator it = _iterator(l);
    while(_hasNext(it)) {
        BasicObject ob = _next(it);
        puts(_toString(ob));
    }

    printf("%d\n", _size(l));

    puts(_toString(l));

    BasicObject ba = _removeIndex(l,2);
    _destroy(ba);
    ba = _removeIndex(l,2);
    _destroy(ba);

    puts(_toString(l));

    String s = NULL;
    s = _getIndex(l,0);
    puts(_toString(s));
    s = _getIndex(l,1);
    puts(_toString(s));
    s = _getIndex(l,1);
    puts(_toString(s));

    puts("**************");
    puts(_toString(l));
    puts("**************");

    printf("HC: %d\n",_hashcode(l));

    List l1 = createList();
    _add(l1,newString("hola1"));
    _add(l1,newString("hola2"));
    _add(l1,newString("hola3"));
    _add(l1,newString("hola4"));
    _add(l1,newString("hola4"));
```

```
    _add(l1,newString("hola5"));
    printf("HC: %d\n",_hashcode(l1));
    printf("l1="); _println(l1);


    List l2 = createList();
    _add(l2,newString("hola1"));
    _add(l2,newString("hola2"));
    _add(l2,newString("hola3"));
    _add(l2,newString("hola4"));
    _add(l2,newString("hola4"));
    _add(l2,newString("hola5"));

    printf("HC: %d\n", _hashcode(l2));
    printf("l2="); _println(l2);


    Collection c = _toCollection(l2);

   printf("c=%s\n",_toString(c));

   printf("l1.equals(c)=%s\n",_bToString(_equals(l1,c)));
   _removeIndex(l2,5);
   _removeIndex(l2,4);
   puts("Eliminamos de l2 hola5 y hola4");
   printf("l2=%s\n",_toString(l2));
   printf("c=%s\n",_toString(c));
   puts("Añadimos a c hola4 y hola5");
   _add(c,newString("hola4"));
   _add(c,newString("hola5"));
   printf("l1=%s\n",_toString(l1));
   printf("l2=%s\n",_toString(l2));
   printf("c=%s\n",_toString(c));
   printf("l1.equals(c)=%s\n",_bToString(_equals(l1,c)));

   puts("Elimnamos hola5");
   _remove(c,newString("hola5"));
   printf("l1=%s\n",_toString(l1));
   printf("l2=c=%s\n",_toString(l2));
   printf("c=l2=%s\n",_toString(c));
   printf("l1.equals(c)=%s\n",_bToString(_equals(l1,c)));

   _gc_println(); printf("\n");

   _destroyAll(l1);
   _destroyAll(l2);

   _gc_println(); printf("\n");
   _gc_flush();
   printf("gcflush(): "); _gc_println(); printf("\n");

   fflush(stdout);
}
```

Ejemplo: Creación y uso de Árboles Binarios

```
void testBTree(void) {
      puts("**********************************");
      puts("******* TEST BINARY TREE  *********");
      puts("**********************************");

      _gc_new();
      BinaryTree t = newSimpleEmptyBTree();
      puts(_toString(t));

      t = newSimpleLeaftBTree(newString("poema"));
      puts(_toString(t));

      t = newSimpleBTree(
                  newString("A"),
                  newSimpleLeaftBTree(newString("B")),
                  newSimpleBTree(
                              newString("C"),
                              newSimpleEmptyBTree(),
                              newSimpleEmptyBTree()
                  )
    );
      printf("t:");
      puts(_toString(t));
      printf("size(t):%d\n",_size(t));

      BinaryTree t1 = newSimpleBTree(
                        newString("A"),
                        newSimpleLeaftBTree(newString("B")),
                        newSimpleBTree(
                                    newString("C"),
                                    newSimpleEmptyBTree(),
                                    newSimpleEmptyBTree()
                        )
      );

      printf("t1:%s\n",_toString(t1));
      printf("t1.equals(t):%s\n",_bToString(_equals(t1,t)));

      _setLeftTree(t1,NULL);
      printf("t1:%s\n",_toString(t1));
      printf("t1.equals(t):%s\n",_bToString(_equals(t1,t)));
      printf("size(t1):%d\n",_size(t1));


    _gc_flush();
}
```

Ejemplo: Creación y uso de un Árboles N-Arios

```
void testNAryTree(void) {
	puts("**********************************");
	puts("******* TEST NAry TREE  *********");
	puts("**********************************");

	_gc_new();
	NAryTree t = newSimpleEmptyNAryTree();

	puts(_toString(t));

	t = newSimpleLeaftNAryTree(newString("poema"));
	NAryTree t1 = t;
	puts(_toString(t1));

	t = newSimpleNAryTree(
			NULL,2,
			newSimpleLeaftNAryTree(newString("Rafael")),
			newSimpleLeaftNAryTree(newString("Moneo"))
	);
	NAryTree t2 = t;
	puts(_toString(t2));

	t = newSimpleNAryTree(
			newString("A"),4,
			newSimpleLeaftNAryTree(newString("B")),
			newSimpleNAryTree(
					newString("C"),2,
					newSimpleEmptyNAryTree(),
					newSimpleEmptyNAryTree()
			),
			t1,
			t2
    );
	printf("t:");
	puts(_toString(t));
	printf("size(t):%d\n",_size(t));

	t1 = newSimpleNAryTree(
			newString("A"),4,
			newSimpleLeaftNAryTree(newString("B")),
			newSimpleNAryTree(
					newString("C"),2,
					newSimpleEmptyNAryTree(),
					newSimpleEmptyNAryTree()
			),
			t1,
			t2
    );

	printf("t1:");
	puts(_toString(t1));
	printf("t.equals(t1):%s\n",_bToString(_equals(t,t1)));

	_removeChildren(t1,1);
	printf("t1:");
	puts(_toString(t1));
```

```
      printf("t.equals(t1):%s\n",_bToString(_equals(t,t1)));

    _gc_flush();
}
```

Ejemplo: Creación y uso de Mapas

```
void testHashMap() {
    puts("***************************");
    puts("***** TEST HASH MAP  ****");
    puts("***************************");

    _gc_new();

      Map m = newHashMap();
    _put(m,newString("k1"), newString("v1"));
    _put(m,newString("k2"), newString("v2"));
    _put(m,newString("k2"), newString("v3"));
    _put(m,newString("k3"), newString("v3"));
    _put(m,newString("k4"), newString("v43"));
    _put(m,newString("k5"), newString("v4"));
    _put(m,newString("k6"), newString("v21"));
    _put(m,newString("k7"), newString("v4"));
    _put(m,newString("k8"), newString("v11"));
    _put(m,newString("k9"), newString("v32"));
    _put(m,newString("k10"), newString("v21"));
    _put(m,newString("k11"), newString("v43"));
    _put(m,newString("k12"), newString("v12"));
    _put(m,newString("k13"), newString("v14"));
    _put(m,newString("l19"), newString("encontrado"));


    puts(_toString(m));

    printf("m.contains('k1')=");
    puts(_bToString(_contains(m,newString("k1"))));

    printf("m.contains('k043')=");
    puts(_bToString(_contains(m,newString("k043"))));

    printf("m.get('k3')=");
    BasicObject ob = _get(m,newString("k3"));
    puts(_toString(ob));

    printf("Recorrido entrySet():\n");
    Set es = _entrySet(m);
    Iterator it = _iterator(es);
    while(_hasNext(it)) {
      BasicObject ob = _next(it);
      puts(_toString(ob));
    }
    _destroy(it);

    printf("Recorrido keySet():\n");
    es = _keySet(m);
    it = _iterator(es);
    while(_hasNext(it)) {
      BasicObject ob = _next(it);
```

```
   puts(_toString(ob));
}
_destroy(it);

printf("Recorrido values():\n");
Collection cs = _values(m);
it = _iterator(cs);
while(_hasNext(it)) {
   BasicObject ob = _next(it);
   puts(_toString(ob));
}
_destroy(it);

printf("m.get('119')=");
ob = _get(m,newString("119"));
if (ob!=NULL) puts(_toString(ob));
else puts("VALOR NO ENCONTRADO");

puts(_toString(m));
printf("m.remove(k3)=");
ob = _remove(m,newString("k3"));
puts(_toString(ob));
puts(_toString(m));

printf("m.get('119')=");
ob = _get(m,newString("119"));
if (ob!=NULL) puts(_toString(ob));
else puts("VALOR NO ENCONTRADO");

_clear(m);
puts(_toString(m));

_put(m,newString("k1"), newString("v1"));
_put(m,newString("k2"), newString("v2"));
_put(m,newString("k2"), newString("v3"));
_put(m,newString("k3"), newString("v3"));

printf("m=");
puts(_toString(m));

Map m1 = newHashMap();
_put(m1,newString("k1"), newString("v1"));
_put(m1,newString("k2"), newString("v2"));
_put(m1,newString("k2"), newString("v3"));
_put(m1,newString("k3"), newString("v3"));
printf("m1=");
puts(_toString(m1));

printf("LOS MAPAS (m,m1) SON IGUALES? %s\n", _bToString(_equals(m,m1)));

printf("m1.remove('k2')");
_remove(m1,newString("k2"));
printf("m1=");
puts(_toString(m1));

printf("LOS MAPAS (m,m1) SON IGUALES? %s\n", _bToString(_equals(m,m1)));

_clear(m1);
_put(m1,newString("k1"), newString("v1"));
```

```
    _put(m1,newString("k2"), newString("v2"));
    _put(m1,newString("k2"), newString("v3"));
    _put(m1,newString("k3"), newString("v4"));
    printf("m1=");
    puts(_toString(m1));
    printf("LOS MAPAS (m,m1) SON IGUALES? %s\n", _bToString(_equals(m,m1)));

    _destroy(m);
    _destroy(m1);

    _gc_destroyLast();
}
```

Ejemplo: Creación y uso de Conjuntos

```
void testHashSet(Set (*setFactory)(), const pchar ntest) {

    puts("***************************");
    printf("***** %s ****\n",ntest);
    puts("***************************");

    printf("Hola %s\n",_toString(newString("e2")));
    fflush(stdout);

    _gc_new();

    Set s = setFactory();
    _add(s,newString("e2"));
    _add(s,newString("e5"));
    _add(s,newString("e3"));
    _add(s,newString("e4"));
    _add(s,newString("e1"));
    puts(_toString(s));

    fflush(stdout);

    Set s1 = setFactory();
    _add(s1,newString("e4"));
    _add(s1,newString("e5"));
    _add(s1,newString("e1"));
    _add(s1,newString("e2"));
    _add(s1,newString("e3"));
    puts(_toString(s1));


    printf("LOS CONJUNTOS SON IGUALES? %s\n", _bToString(_equals(s,s1)));

    printf("s1.remove('e5')\n");
    _remove(s1,newString("e5"));
    printf("s1.add('e8')\n");
    _add(s1,newString("e8"));
    puts(_toString(s1)); fflush(stdout);
    printf("LOS CONJUNTOS SON IGUALES? %s\n", _bToString(_equals(s,s1)));
fflush(stdout);

    printf("EXISTE e8? %s\n", _bToString(_contains(s1,newString("e8"))));
fflush(stdout);
```

```
    printf("EXISTE e5? %s\n", _bToString(_contains(s1,newString("e5"))));
fflush(stdout);

    printf("s1.clear()\n");

    _clear(s1);
    puts(_toString(s1)); fflush(stdout);

    _gc_destroyLast();  // RRR
}
```

Ejemplo: Creación y uso de un Grafos

```
void testHashGraph() {
    puts("***************************");
    puts("***** TEST HASH GRAPH  ****");
    puts("***************************");

    Graph g = newHashDirectedGraph();
    String v1 = newString("v1");
    String v2 = newString("v2");
    String v3 = newString("v3");
    String v4 = newString("v4");
    String v5 = newString("v5");
    String v6 = newString("v6");




    _addVertex(g,v1);
    _addVertex(g,v2);
    _addVertex(g,v3);
    _addVertex(g,v4);
    _addVertex(g,v5);
    _addVertex(g,v6);
    puts(_toString(g)); fflush(stdout);

    _addEdge(g,v1,v2,newInt(1));
    _addEdge(g,v2,v5,newInt(2));
    _addEdge(g,v5,v1,newInt(1));
    _addEdge(g,v1,v5,newInt(1));
    _addEdge(g,v3,v4,newInt(1));
    _addEdge(g,v6,v3,newInt(1));


    _println(g);
    puts(_toString(g));

    printf("hasEdge(%s,%s)=%s\n",_toString(v1),_toString(v2),
             _bToString(_hasEdge(g,v1,v2)));
    printf("hasEdge(%s,%s)=%s\n",_toString(v2),_toString(v1),
                   _bToString(_hasEdge(g,v2,v1)));

    _removeVertex(g,v3);
    printf("g.remove(v3)="); _println(g);

    _removeEdge(g,v1,v5);
    printf("g.remove(v1,v5)="); _println(g);
```

```
    _removeEdge(g,v5,v1);
    printf("g.remove(v5,v1)="); _println(g);
}
```

Ejemplo: Creación y uso de Tablas

```
void testHashTable() {
    _gc_new();
       puts("****************************************");
          puts("***** TEST HASH TABLE ****");
          puts("****************************************");
      Table t = newHashTable();
      _put2k(t,newString("r6"),newString("c5"),newString("v1"));
      _put2k(t,newString("r1"),newString("c3"),newString("v2"));
      _put2k(t,newString("r1"),newString("c2"),newString("v3"));
      _put2k(t,newString("r2"),newString("c2"),newString("v4"));
      _put2k(t,newString("r4"),newString("c4"),newString("v5"));
      _put2k(t,newString("r4"),newString("c1"),newString("v6"));
      _put2k(t,newString("r4"),newString("c3"),newString("v7"));
      _put2k(t,newString("r4"),newString("c5"),newString("v8"));
      _put2k(t,newString("r5"),newString("c3"),newString("v9"));
      _put2k(t,newString("r5"),newString("c5"),newString("v10"));
      _put2k(t,newString("r3"),newString("c5"),newString("v11"));
      _put2k(t,newString("r5"),newString("c5"),newString("v12"));
      _put2k(t,newString("r5"),newString("c5"),newString("v13"));

      puts("Table t:");
      puts("*****************");
      _println(t);

      Set s = NULL;
      BasicObject bo = NULL;

      Map m = _row(t,newString("r1"));
    printf("t.row('r1')=%s\n",_toString(m));
    m = _row(t,newString("r2"));
    printf("t.row('r2')=%s\n",_toString(m));
    s = _keySet(m);
    printf("t.row('r2').keySet()=%s\n",_toString(s));
    bo = _get(m,newString("c2"));
    printf("t.row('r2').get('c2')=%s\n",_toString(bo));


    m = _row(t,newString("r1"));
    s = _keySet(m);
    printf("t.row('r1').keySet()=%s\n",_toString(s));
    s = _entrySet(m);
    printf("t.row('r1').entrySet()=%s\n",_toString(s));

    bo = _get(m,newString("c2"));
    printf("t.row('r1').get('c2')=%s\n",_toString(bo));
    printf("t.row('r1')=%s\n",_toString(m));

    m = _row(t,newString("r4"));
    bo = _get(m,newString("c3"));
    printf("t.row('r4').get('c3')=%s\n",_toString(bo));

    printf("print(t):\n");
```

```
   puts(_toString(t));
   _remove2k(t,newString("r5"),newString("c3"));
printf("t.remove('r5','c3')\n");fflush(stdout);
printf("print(t):\n");fflush(stdout);
   puts(_toString(t));fflush(stdout);
   _remove2k(t,newString("r5"),newString("c5"));
printf("t.remove('r5','c5')\n"); fflush(stdout);
printf("print(t):\n");fflush(stdout);

   puts(_toString(t)); fflush(stdout);


   bo = _get2k(t,newString("r4"),newString("c5"));
   printf("(r4,c5):%s\n",_toString(bo));
   bo = _get2k(t,newString("r5"),newString("c5"));
   if (bo==NULL) {
     printf("(r5,c5) No existe.\n");
   } else {
      printf("(r5,c5):%s\n",_toString(bo));
   }

   fflush(stdout);

puts("Utilizacion de iteradores:");
Set s1 = _rowKeySet(t);
   Iterator it = _iterator(s1);

   String sb = newString("");
   while(_hasNext(it)) {
          BasicObject r = _next(it);
       Map rows = _row(t,r);
       //puts(rows->toString(rows));
       Set s2 = _keySet(rows);
       Iterator it2 = _iterator(s2);
       while(_hasNext(it2)) {
           BasicObject c = _next(it2);
           BasicObject v = _get(rows,c);
             //puts(v->toString(v));
           _set(sb,"(");
           _concat(sb,_toString(r));
           _concat(sb,", ");
           _concat(sb,_toString(c));
           _concat(sb,")->");
           _concat(sb,_toString(v));
           puts(_toString(sb));
          // puts(v->toString(v));
       }
   }

   puts("*******************************");
   puts("******* PPPPP  PPPPP  *********");
   puts("*******************************");
   Set sc = _cellSet(t);
   it = _iterator(sc);
   while(_hasNext(it)) {
          BasicObject ob = (BasicObject)_next(it);
          puts(_toString(ob));
   }
```

```
    Table t1 = newHashTable();
    _put2k(t1,newString("r6"),newString("c5"),newString("v1"));
    _put2k(t1,newString("r1"),newString("c3"),newString("v2"));
    _put2k(t1,newString("r1"),newString("c2"),newString("v3"));
    _put2k(t1,newString("r2"),newString("c2"),newString("v4"));

    Table t2 = newHashTable();
    _put2k(t2,newString("r6"),newString("c5"),newString("v1"));
    _put2k(t2,newString("r1"),newString("c3"),newString("v2"));
    _put2k(t2,newString("r1"),newString("c2"),newString("v3"));
    _put2k(t2,newString("r2"),newString("c2"),newString("v4"));

    printf("t1=%s\n",_toString(t1)); fflush(stdout);
    printf("t2=%s\n",_toString(t2)); fflush(stdout);
  printf("t1.equals(t2)=%s\n",_bToString(_equals(t1,t2))); fflush(stdout);

    _put2k(t2,newString("r2"),newString("c2"),newString("v5"));
    printf("t2.put('r2','c2','v5')\n");
    printf("t2=%s\n",_toString(t2));
  printf("t1.equals(t2)=%s\n",_bToString(_equals(t1,t2)));
  _remove2k(t2,newString("r2"),newString("c2"));
  printf("t2.remove('r2','c2')\n");
  printf("t2=%s\n",_toString(t2));
  printf("t1.equals(t2)=%s\n",_bToString(_equals(t1,t2)));
    _gc_destroyLast();
}
```