```
library Table /* made by Bribe, special thanks to Vexorian & Nestharus, version 5.0.0.0

    One map, one hashtable. Welcome to NewTable.

    This latest version of Table introduces a new struct: HashTableEx.
    This behaves like a HashTable but has some additional functionality
    such as storing each saved index in order to allow iteration and
    automatic destruction.

    API

    ------------
    struct Table
    | static method create takes nothing returns Table
    |     create a new Table
    |
    | method destroy takes nothing returns nothing
    |     destroy it
    |
    | method flush takes nothing returns nothing
    |     flush all stored values inside of it
    |
    | method remove takes integer key returns nothing
    |     remove the value at index "key"
    |
    | method operator []= takes integer key, $TYPE$ value returns nothing
    |     assign "value" to index "key"
    |
    | method operator [] takes integer key returns $TYPE$
    |     load the value at index "key"
    |
    | method has takes integer key returns boolean
    |     whether or not the key was assigned
    |
    ----------------
    struct TableArray
    | static method operator [] takes integer array_size returns TableArray
    |     create a new array of Tables of size "array_size"
    |
    | method destroy takes nothing returns nothing
    |     destroy it
    |
    | method flush takes nothing returns nothing
    |     flush and destroy it
    |
    | method operator size takes nothing returns integer
    |     returns the size of the TableArray
    |
    | method operator [] takes integer key returns Table
    |     returns a Table accessible exclusively to index "key"
*/

globals
    private integer less = 0    //Index generation for TableArrays (below 0).
    private integer more = 8190 //Index generation for Tables.
    //Configure it if you use more than 8190 "key" variables in your map (this will never happen though).

    private hashtable ht = InitHashtable()
    private key sizeK
    private key listK
endglobals

private struct dex extends array
    static method operator size takes nothing returns Table
        return sizeK
    endmethod
    static method operator list takes nothing returns Table
        return listK
    endmethod
endstruct

private struct handles extends array
    method operator []= takes integer key, handle h returns nothing
        if h != null then
            call SaveFogStateHandle(ht, this, key, ConvertFogState(GetHandleId(h)))
        elseif HaveSavedHandle(ht, this, key) then
            call RemoveSavedHandle(ht, this, key)
        endif
    endmethod
    method has takes integer key returns boolean
```

```
        return HaveSavedHandle(ht, this, key)
    endmethod
    method remove takes integer key returns nothing
        call RemoveSavedHandle(ht, this, key)
    endmethod
endstruct

private struct agents extends array
    method operator []= takes integer key, agent value returns nothing
        call SaveAgentHandle(ht, this, key, value)
    endmethod
endstruct

//! textmacro NEW_ARRAY_BASIC takes SUPER, FUNC, TYPE
private struct $TYPE$s extends array
    method operator [] takes integer key returns $TYPE$
        return Load$FUNC$(ht, this, key)
    endmethod
    method operator []= takes integer key, $TYPE$ value returns nothing
        call Save$FUNC$(ht, this, key, value)
    endmethod
    method has takes integer key returns boolean
        return HaveSaved$SUPER$(ht, this, key)
    endmethod
    method remove takes integer key returns nothing
        call RemoveSaved$SUPER$(ht, this, key)
    endmethod
endstruct
private module $TYPE$m
    method operator $TYPE$ takes nothing returns $TYPE$s
        return this
    endmethod
endmodule
//! endtextmacro

//! textmacro NEW_ARRAY takes FUNC, TYPE
private struct $TYPE$s extends array
    method operator [] takes integer key returns $TYPE$
        return Load$FUNC$Handle(ht, this, key)
    endmethod
    method operator []= takes integer key, $TYPE$ value returns nothing
        call Save$FUNC$Handle(ht, this, key, value)
    endmethod
    method has takes integer key returns boolean
        return HaveSavedHandle(ht, this, key)
    endmethod
    method remove takes integer key returns nothing
        call RemoveSavedHandle(ht, this, key)
    endmethod
endstruct
private module $TYPE$m
    method operator $TYPE$ takes nothing returns $TYPE$s
        return this
    endmethod
endmodule
//! endtextmacro

//Run these textmacros to include the entire hashtable API as wrappers.
//Don't be intimidated by the number of macros - Vexorian's map optimizer is
//supposed to kill functions which inline (all of these functions inline).
//! runtextmacro NEW_ARRAY_BASIC("Real", "Real", "real")
//! runtextmacro NEW_ARRAY_BASIC("Boolean", "Boolean", "boolean")
//! runtextmacro NEW_ARRAY_BASIC("String", "Str", "string")
//New textmacro to allow table.integer[] syntax for compatibility with textmacros that might desire it.
//! runtextmacro NEW_ARRAY_BASIC("Integer", "Integer", "integer")

//! runtextmacro NEW_ARRAY("Player", "player")
//! runtextmacro NEW_ARRAY("Widget", "widget")
//! runtextmacro NEW_ARRAY("Destructable", "destructable")
//! runtextmacro NEW_ARRAY("Item", "item")
//! runtextmacro NEW_ARRAY("Unit", "unit")
//! runtextmacro NEW_ARRAY("Ability", "ability")
//! runtextmacro NEW_ARRAY("Timer", "timer")
//! runtextmacro NEW_ARRAY("Trigger", "trigger")
//! runtextmacro NEW_ARRAY("TriggerCondition", "triggercondition")
//! runtextmacro NEW_ARRAY("TriggerAction", "triggeraction")
//! runtextmacro NEW_ARRAY("TriggerEvent", "event")
//! runtextmacro NEW_ARRAY("Force", "force")
//! runtextmacro NEW_ARRAY("Group", "group")
//! runtextmacro NEW_ARRAY("Location", "location")
//! runtextmacro NEW_ARRAY("Rect", "rect")
//! runtextmacro NEW_ARRAY("BooleanExpr", "boolexpr")
//! runtextmacro NEW_ARRAY("Sound", "sound")
//! runtextmacro NEW_ARRAY("Effect", "effect")
//! runtextmacro NEW_ARRAY("UnitPool", "unitpool")
//! runtextmacro NEW_ARRAY("ItemPool", "itempool")
//! runtextmacro NEW_ARRAY("Quest", "quest")
//! runtextmacro NEW_ARRAY("QuestItem", "questitem")
//! runtextmacro NEW_ARRAY("DefeatCondition", "defeatcondition")
//! runtextmacro NEW_ARRAY("TimerDialog", "timerdialog")
//! runtextmacro NEW_ARRAY("Leaderboard", "leaderboard")
//! runtextmacro NEW_ARRAY("Multiboard", "multiboard")
//! runtextmacro NEW_ARRAY("MultiboardItem", "multiboarditem")
//! runtextmacro NEW_ARRAY("Trackable", "trackable")
//! runtextmacro NEW_ARRAY("Dialog", "dialog")
//! runtextmacro NEW_ARRAY("Button", "button")
//! runtextmacro NEW_ARRAY("TextTag", "texttag")
//! runtextmacro NEW_ARRAY("Lightning", "lightning")
//! runtextmacro NEW_ARRAY("Image", "image")
//! runtextmacro NEW_ARRAY("Ubersplat", "ubersplat")
//! runtextmacro NEW_ARRAY("Region", "region")
//! runtextmacro NEW_ARRAY("FogState", "fogstate")
//! runtextmacro NEW_ARRAY("FogModifier", "fogmodifier")
//! runtextmacro NEW_ARRAY("Hashtable", "hashtable")

struct Table extends array

    // Implement modules for intuitive syntax (tb.handle; tb.unit; etc.)
    implement realm
    implement integerm
    implement booleanm
    implement stringm
    implement playerm
    implement widgetm
    implement destructablem
    implement itemm
    implement unitm
    implement abilitym
    implement timerm
    implement triggerm
    implement triggerconditionm
    implement triggeractionm
    implement eventm
    implement forcem
    implement groupm
    implement locationm
    implement rectm
    implement boolexprm
    implement soundm
    implement effectm
    implement unitpoolm
    implement itempoolm
    implement questm
    implement questitemm
    implement defeatconditionm
    implement timerdialogm
    implement leaderboardm
    implement multiboardm
    implement multiboarditemm
    implement trackablem
    implement dialogm
    implement buttonm
    implement texttagm
    implement lightningm
    implement imagem
    implement ubersplatm
    implement regionm
    implement fogstatem
    implement fogmodifierm
    implement hashtablem

    method operator handle takes nothing returns handles
        return this
    endmethod

    method operator agent takes nothing returns agents
        return this
    endmethod

    //set this = tb[GetSpellAbilityId()]
    method operator [] takes integer key returns Table
        return LoadInteger(ht, this, key) //return this.integer[key]
    endmethod

    //set tb[389034] = 8192
    method operator []= takes integer key, Table tb returns nothing
        call SaveInteger(ht, this, key, tb) //set this.integer[key] = tb
    endmethod

    //set b = tb.has(2493223)
    method has takes integer key returns boolean
        return HaveSavedInteger(ht, this, key) //return this.integer.has(key)
    endmethod

    //call tb.remove(294080)
    method remove takes integer key returns nothing
        call RemoveSavedInteger(ht, this, key) //call this.integer.remove(key)
    endmethod

    //Remove all data from a Table instance
    method flush takes nothing returns nothing
        call FlushChildHashtable(ht, this)
    endmethod

    //local Table tb = Table.create()
    static method create takes nothing returns Table
        local Table this = dex.list[0]

        if this == 0 then
            set this = more + 1
            set more = this
        else
            set dex.list[0] = dex.list[this]
            call dex.list.remove(this) //Clear hashed memory
        endif

        debug set dex.list[this] = -1
        return this
    endmethod

    // Removes all data from a Table instance and recycles its index.
    //
    //     call tb.destroy()
    //
    method destroy takes nothing returns nothing
        debug if dex.list[this] != -1 then
            debug call BJDebugMsg("Table Error: Tried to double-free instance: " + I2S(this))
            debug return
        debug endif

        call this.flush()

        set dex.list[this] = dex.list[0]
        set dex.list[0] = this
    endmethod

    //! runtextmacro optional TABLE_BC_METHODS()
endstruct

//! runtextmacro optional TABLE_BC_STRUCTS()

struct TableArray extends array

    //Returns a new TableArray to do your bidding. Simply use:
    //
    //    local TableArray ta = TableArray[array_size]
    //
    static method operator [] takes integer array_size returns TableArray
        local Table tb = dex.size[array_size] //Get the unique recycle list for this array size
        local TableArray this = tb[0]         //The last-destroyed TableArray that had this array size

        debug if array_size <= 0 then
            debug call BJDebugMsg("TypeError: Invalid specified TableArray size: " + I2S(array_size))
            debug return 0
        debug endif

        if this == 0 then
            set this = less - array_size
            set less = this
        else
            set tb[0] = tb[this]  //Set the last destroyed to the last-last destroyed
            call tb.remove(this)  //Clear hashed memory
        endif
        set dex.size[this] = array_size //This remembers the array size
        return this
    endmethod

    //Returns the size of the TableArray
    method operator size takes nothing returns integer
        return dex.size[this]
    endmethod

    //This magic method enables two-dimensional[array][syntax] for Tables,
    //similar to the two-dimensional utility provided by hashtables them-
    //selves.
    //
    //ta[integer a].unit[integer b] = unit u
    //ta[integer a][integer c] = integer d
    //
    //Inline-friendly when not running in debug mode
    //
    method operator [] takes integer key returns Table
        static if DEBUG_MODE then
            local integer i = this.size
            if i == 0 then
                call BJDebugMsg("IndexError: Tried to get key from invalid TableArray instance: " + I2S(this))
                return 0
            elseif key < 0 or key >= i then
                call BJDebugMsg("IndexError: Tried to get key [" + I2S(key) + "] from outside TableArray bounds: " + I2S(i))
                return 0
            endif
        endif
        return this + key
    endmethod

    //Destroys a TableArray without flushing it; I assume you call .flush()
    //if you want it flushed too. This is a public method so that you don't
    //have to loop through all TableArray indices to flush them if you don't
    //need to (ie. if you were flushing all child-keys as you used them).
    //
    method destroy takes nothing returns nothing
        local Table tb = dex.size[this.size]

        debug if this.size == 0 then
            debug call BJDebugMsg("TypeError: Tried to destroy an invalid TableArray: " + I2S(this))
            debug return
        debug endif

        if tb == 0 then
            //Create a Table to index recycled instances with their array size
            set tb = Table.create()
            set dex.size[this.size] = tb
        endif

        call dex.size.remove(this) //Clear the array size from hash memory

        set tb[this] = tb[0]
        set tb[0] = this
    endmethod

    private static Table tempTable
    private static integer tempEnd

    //Avoids hitting the op limit
    private static method clean takes nothing returns nothing
        local Table tb = .tempTable
        local integer end = tb + 0x1000
        if end < .tempEnd then
            set .tempTable = end
            call ForForce(bj_FORCE_PLAYER[0], function thistype.clean)
        else
            set end = .tempEnd
        endif
        loop
            call tb.flush()
            set tb = tb + 1
            exitwhen tb == end
        endloop
    endmethod

    //Flushes the TableArray and also destroys it. Doesn't get any more
    //similar to the FlushParentHashtable native than this.
    //
    method flush takes nothing returns nothing
        debug if this.size == 0 then
            debug call BJDebugMsg("TypeError: Tried to flush an invalid TableArray instance: " + I2S(this))
            debug return
        debug endif
        set .tempTable = this
        set .tempEnd = this + this.size
        call ForForce(bj_FORCE_PLAYER[0], function thistype.clean)
        call this.destroy()
    endmethod

endstruct

//Added in Table 4.0. A fairly simple struct but allows you to do more
//than that which was previously possible.
struct HashTable extends array

    //Enables myHash[parentKey][childKey] syntax.
    //Basically, it creates a Table in the place of the parent key if
    //it didn't already get created earlier.
    method operator [] takes integer index returns Table
        local Table t = Table(this)[index]
        if t == 0 then
            set t = Table.create()
            set Table(this)[index] = t
        endif
        return t
    endmethod

    //You need to call this on each parent key that you used if you
    //intend to destroy the HashTable or simply no longer need that key.
    method remove takes integer index returns nothing
        local Table t = Table(this)[index]
        if t != 0 then
            call t.destroy()               //clear indexed table
            call Table(this).remove(index) //clear reference to that table
        endif
    endmethod

    //Added in version 4.1
    method has takes integer index returns boolean
        return Table(this).has(index)
    endmethod

    //HashTables are mostly just fancy Table indices.
    method destroy takes nothing returns nothing
        call Table(this).destroy()
    endmethod

    static method create takes nothing returns thistype
        return Table.create()
    endmethod

endstruct

//Added in Table 5.0. Similar to the HashTable struct but with the
//ability to log each value saved into the HashTable to automate
//deallocation.
private module TRACKER
    static thistype tracker = 0
    private static method onInit takes nothing returns nothing
        set tracker = Table.create()
    endmethod
endmodule
struct HashTableEx extends array

    implement TRACKER

    method operator [] takes integer index returns Table
        local integer i
        local Table t = Table(this)[index]
        if t == 0 then
            set t = Table.create()
            set Table(this)[index] = t
            set t = tracker[this]          //get the tracking table's index for this HashTable
            set i = t[0] + 1               //increase its size
            set t[0] = i                   //save that size
            set t[i] = index               //index the user's index to the tracker's slot at 'size'
        endif
        return t
    endmethod
    //Extremely inefficient, but gets the job done if needed.
    method remove takes integer index returns nothing
        local integer i
        local integer j
        local Table t = Table(this)[index]
        if t != 0 then
            call t.destroy()               //clear indexed table
            call Table(this).remove(index) //clear reference to that table
            set t = tracker[this]
            set i = t[0]
            set j = i
            loop
                exitwhen t[i] == index //removal is o(n) based
                set i = i - 1
            endloop
            if i < j then
                set t[i] = t[j] //pop last item in the stack and insert in place of this removed item
            endif
            call t.remove(j) //free reference to the index
            set t[0] = j - 1 //decrease size of stack
        endif
    endmethod

    method has takes integer index returns boolean
        return Table(this).has(index)
    endmethod

    //Useful for debugging purposes I suppose.
    //Treats the HashTable like a TableArray when used instead of [].
    method getIndex takes integer i returns Table
        return tracker[this][i]
    endmethod

    method destroy takes nothing returns nothing
        local Table t = tracker[this] //tracker table
        local Table t2                //sub-tables of the primary HashTable
        local integer i = t[0]        //get the number of tracked indices
        loop
            exitwhen i == 0
            set t2 = t[i]
            call t2.destroy()           //clear indexed sub-table
            call Table(this).remove(t2) //clear reference to sub-table
            set i = i - 1
        endloop
        call t.destroy()           //clear tracking sub-table
        call tracker.remove(this)  //clear reference to that table
        call Table(this).destroy()
    endmethod

    static method create takes nothing returns thistype
        local thistype this = Table.create()
        set tracker[this][0] = 0
        return this
    endmethod

endstruct

endlibrary
```