```
unit altpointers;

interface


uses
  W3C.TypedArray,
  System.Types,
  System.Types.Convert,
  System.Memory,
  system.memory.allocation,
  System.Memory.Buffer,
  System.Memory.Views;


type

  Pointer = variant;

  TPointerData = record
    Offset: integer;
    Buffer: JArrayBuffer;
    View:   JUint8Array;
  end;

function IncPointer(Src: Pointer; AddValue: integer): Pointer;
function DecPointer(Src: Pointer; DecValue: integer): Pointer;
function EquPointer(src, dst : Pointer): boolean;

// a := a + bytes
operator + (Pointer,   integer): Pointer uses IncPointer;

// a := a - bytes
operator - (Pointer,   integer): Pointer uses DecPointer;

// if a = b then
operator = (Pointer,   Pointer): boolean uses EquPointer;

function  Allocmem(const Size: integer): Pointer;
function  Addr(const Source: Pointer; const Offset: integer): Pointer;
procedure FreeMem(const Source: Pointer);
procedure MemSet(const Target: pointer; const Value: byte); overload;
procedure MemSet(const Target: pointer; const Values: array of byte); overload;
function  MemGet(const Source: pointer): byte; overload;
function  MemGet(const Source: pointer; ReadLength: integer): TByteArray; overload;


implementation
```

```
function MemGet(const Source: pointer): byte;
begin
  if (Source) then
  begin
    var SrcData: TPointerData;
    asm @SrcData = @Source; end;
    result := SrcData.View.items[SrcData.Offset];
  end else
  raise Exception.Create('MemGet failed, invalid pointer error');
end;

function MemGet(const Source: pointer; ReadLength: integer): TByteArray;
begin
  if (Source) then
  begin
    var SrcData: TPointerData;
    asm @SrcData = @Source; end;

    var Offset := SrcData.Offset;

    while ReadLength > 0 do
    begin
      result.add( SrcData.View.items[Offset] );
      inc(Offset);
      dec(ReadLength);

      if offset >= SrcData.View.byteLength then
        raise Exception.Create('MemGet failed, offset exceeds memory');
    end;
  end else
  raise Exception.Create('MemGet failed, invalid pointer error');
end;

procedure MemSet(const Target: pointer; const Value: byte);
begin
  var DstData: TPointerData;
  asm @DstData = @Target; end;
  dstData.View.items[DstData.Offset] := value;
end;

procedure MemSet(const Target: pointer; const Values: array of byte);
begin
  if Values.length > 0 then
  begin
    var DstData: TPointerData;
    asm @DstData = @Target; end;
```

```
    var offset := DstData.Offset;
    for var x := low(Values) to high(Values) do
    begin
      dstData.View.items[offset] := Values[x];
      inc(offset);
      if offset >= DstData.View.byteLength then
        raise Exception.Create('MemSet failed, offset exceeds memory');
    end;
  end;
end;

function EquPointer(src, dst : Pointer): boolean;
begin
  if (src) then
  begin
    if (dst) then
    begin
      var SrcData: TPointerData;
      var DstData: TPointerData;
      asm @SrcData = @Src; end;
      asm @DstData = @dst; end;
      result := SrcData.buffer = dstData.buffer;
    end;
  end;
end;

function IncPointer(Src: Pointer; AddValue: integer): Pointer;
begin
  if (Src) then
  begin
    // Check that there is an actual change.
    // If not, just return the same pointer
    if AddValue > 0 then
    begin
      // Map source data
      var SrcData: TPointerData;
      asm @SrcData = @Src; end;

      // Calculate new offset, using the current view
      // position as the present location.
      var NewOffset := srcData.Offset;
      inc(NewOffset, AddValue);

      // Make sure the new offset is within the range of the
      // memory buffer. Picky yes, but this is not native land
      if  (NewOffset >=0)
      and (NewOffset < srcData.View.byteLength) then
      begin
```

```
        // Setup new Pointer data
        var Data: TPointerData;
        Data.Buffer := SrcData.Buffer;
        Data.View := SrcData.View;
        Data.Offset := NewOffset;

        // return new pointer
        asm
          @result = @data;
        end;
      end else
      raise Exception.Create('IncPointer failed, offset exceeds memory');
    end else
    result := src;
  end else
  raise Exception.Create('IncPointer failed, invalid pointer error');
end;

function DecPointer(Src: Pointer; DecValue: integer): Pointer;
begin
  if (Src) then
  begin
    // Check that there is an actual change.
    // If not, just return the same pointer
    if DecValue > 0 then
    begin
      // Map source data
      var SrcData: TPointerData;
      asm @SrcData = @Src; end;

      // Calculate new offset, using the current view
      // position as the present location.
      var NewOffset := srcData.Offset;
      dec(NewOffset, DecValue);

      // Make sure the new offset is within the range of the
      // memory buffer. Picky yes, but this is not native land
      if  (NewOffset >=0)
      and (NewOffset < srcData.View.byteLength) then
      begin
        // Setup new Pointer data
        var Data: TPointerData;
        Data.Buffer := SrcData.Buffer;
        Data.View := SrcData.View;
        Data.Offset := NewOffset;

        // return new pointer
        asm
```

```
        @result = @data;
      end;
    end else
    raise Exception.Create('IncPointer failed, offset exceeds memory');
  end else
  result := src;
 end else
 raise Exception.Create('IncPointer failed, invalid pointer error');
end;


function Allocmem(const Size: integer): Pointer;
begin
 if Size > 0 then
 begin
  var Data: TPointerData;
  Data.Offset := 0;
  Data.Buffer := JArrayBuffer.Create(Size);
  Data.View := JUint8Array.Create(Data.Buffer, 0, Size);
  asm
    @result = @data;
  end;
 end else
 raise Exception.Create('Allocmem failed, invalid size error');
end;

function Addr(const Source: Pointer; const Offset: integer): Pointer;
begin
 if (Source) then
 begin
  if offset > 0 then
  begin
    // Map source data
    var SrcData: TPointerData;
    asm @SrcData = @Source; end;

    // Check that offset is valid
    if (Offset >=0) and (offset < srcData.buffer.byteLength) then
    begin
      // Setup new Pointer data
      var Data: TPointerData;
      Data.Buffer := SrcData.Buffer;
      Data.View := SrcData.View;
      Data.Offset := Offset;
      asm
        @result = @data;
      end;
    end else
```

```
      raise Exception.Create('Addr failed, offset exceeds memory');
    end else
    raise Exception.Create('Addr failed, invalid offset error');
  end else
  raise Exception.Create('Addr failed, invalid pointer error');
end;

procedure FreeMem(const Source: Pointer);
begin
  if (source) then
  begin
    // Map source data
    var SrcData: TPointerData;
    asm @SrcData = @Source; end;

    // Flush reference and let the GC take care of it
    SrcData.Buffer := nil;
    SrcData.View := nil;
    SrcData.Offset := 0;
    asm
      srcData = {}
    end;
  end else
  raise Exception.Create('FreeMem failed, invalid pointer error');
end;

end.
```