Aufgabe 1 - „Behälter“ Vector als ADT

**Lösungsidee:**wie bei dem beispiel wo die größe des Felds erst zur Laufzeit fixiert wird ein array auf die gleiche Weise erstellen allerdings immer wenn die größe überschritten werden sollte, wird die funktion GrowVector aufgerufen, die die capacity verdopellt und somit auch den allokierten speicher und die werte des allten vectors in den neuen kopiert.  
  
eine mögliche verbesserung meines codes wäre den vector wieder zu verkleinern wenn der count kleiner als die hälfte von capacity ist, oder den vector nicht jedesmal ums doppelte zu vergrößern sondern eine bessere gewählten wert zu verwenden. Aber das ist immer usecase spezifisch, je nachdem was man mit dem vector anfangen will.

**Zeitaufwand: ~**1h 30min

**Code:**

unit Vector;

interface

const

  MAX\_CAPACITY = MaxInt;

type

  PIntArray = ^IntArray;

  IntArray = array[0..MAX\_CAPACITY] of Integer; // max array size is MaxInt (32767)

  Vec = record

    data: PIntArray;

    count: Integer;

    capacity: Integer;

  end;

procedure InitVector(var v: Vec);

procedure DisposeVector(var v: Vec);

procedure Add(var v: Vec; val: Integer);

procedure SetElementAt(var v: Vec; pos: Integer; val: Integer);

function ElementAt(v: Vec; pos: Integer): Integer;

procedure RemoveElementAt(var v: Vec; pos: Integer);

function Size(v: Vec): Integer;

function Capacity(v: Vec): Integer;

implementation

uses

  SysUtils;

procedure InitVector(var v: Vec);

begin

  v.count := 0;

  v.capacity := 1;

  GetMem(v.data, v.capacity \* SizeOf(Integer));

  if v.data = nil then

  begin

    WriteLn('Error: Heap overflow.');

    Halt(1);

  end;

end;

procedure DisposeVector(var v: Vec);

begin

  FreeMem(v.data, v.capacity \* SizeOf(Integer));

end;

procedure GrowVector(var v: vec);

var

  newCapacity: Integer;

  newData: PIntArray;

  i: Integer;

begin

  newCapacity := v.capacity \* 2;

  if newCapacity > MAX\_CAPACITY then

  begin

    WriteLn('Error: Capacity exceeds maximum value.');

    Halt(1);

  end;

  GetMem(newData, newCapacity \* SizeOf(Integer));

  for i := 0 to v.count - 1 do

    newData^[i] := v.data^[i];

  FreeMem(v.data, v.capacity \* SizeOf(Integer));

  v.data := newData;

  v.capacity := newCapacity;

end;

procedure Add(var v: Vec; val: Integer);

begin

  if v.count = v.capacity then

    GrowVector(v);

  v.data^[v.count] := val;

  Inc(v.count);

end;

procedure SetElementAt(var v: Vec; pos: Integer; val: Integer);

begin

  if (pos < 0) or (pos >= v.count) then

  begin

    WriteLn('Error: Index out of range.');

    Halt(1);

  end;

  v.data^[pos] := val;

end;

function ElementAt(v: Vec; pos: Integer): Integer;

begin

  if (pos < 0) or (pos >= v.count) then

  begin

    WriteLn('Error: Index out of range.');

    Halt(1);

  end;

  ElementAt := v.data^[pos];

end;

procedure RemoveElementAt(var v: Vec; pos: Integer);

var

  i: Integer;

begin

  if (pos < 0) or (pos >= v.count) then

  begin

    WriteLn('Error: Index out of range.');

    Halt(1);

  end;

  for i := pos to v.count - 2 do

    v.data^[i] := v.data^[i + 1];

  Dec(v.count);

end;

function Size(v: Vec): Integer;

begin

  Size := v.count;

end;

function Capacity(v: Vec): Integer;

begin

  Capacity := v.capacity;

end;

end.

**Test Code:**

program VectorTests;

uses Vector;

var

  v: Vec;

  i: Integer;

begin

  // Initialize an empty vector

  InitVector(v);

  // Test adding elements

  WriteLn('Test adding elements:');

  writeln('0 elements - size: ', Size(v), ', capacity: ', Capacity(v));

  Add(v, 1);

  writeln('1 element - size: ', Size(v), ', capacity: ', Capacity(v));

  Add(v, 2);

  writeln('2 elements - size: ', Size(v), ', capacity: ', Capacity(v));

  Add(v, 3);

  writeln('3 elements - size: ', Size(v), ', capacity: ', Capacity(v));

  Add(v, 4);

  writeln('4 elements - size: ', Size(v), ', capacity: ', Capacity(v));

  Add(v, 5);

  writeln('5 elements - size: ', Size(v), ', capacity: ', Capacity(v));

  WriteLn;

  WriteLn;

  // Test getting and setting elements

  WriteLn('Test getting and setting elements:');

  WriteLn('Element at position 2: ', ElementAt(v, 2)); // should print 3

  SetElementAt(v, 2, 6);

  WriteLn('Element at position 2 after setting it to 6: ', ElementAt(v, 2)); // should print 6

  WriteLn;

  WriteLn;

  // Test removing elements

  WriteLn('Test removing elements:');

  WriteLn('Vector size before removing an element: ', Size(v)); // should print 5

  RemoveElementAt(v, 2);

  WriteLn('Vector size after removing an element: ', Size(v)); // should print 4

  WriteLn('Element at position 2 after removing element at position 2: ', ElementAt(v, 2)); // should print 4

  WriteLn;

  WriteLn;

  // Test disposing of vector

  DisposeVector(v);

end.

**Test Ausgabe:**

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung