1 Поиск элементов в массиве

```
function Find<T>(a: array of T;
                 x: T): integer;
begin
  Result := -1;
  for var i:= 0 to a.Length-1 do
    if a[i] = x then
    begin
      Result := i;
      break;
    end;
end;

function FindWhile<T>(a: array of T;
                      x: T): integer;
begin
  var n := a.Length;
  var i := 0;
  while (i < n) and (a[i] <> x) do
    i += 1;
  if i = n then
    Result := -1
  else
    Result := i;
end;

function FindBarrier<T>(a: array of T;
              n: integer; x: T): integer;
begin
  Assert((0 < n) and (n < a.Length));
  a[n] := x;
  var i := 0;
  while a[i] <> x do
    i += 1;
  if i = n then
    Result := -1
  else
    Result := i;
end;

function BinarySearch(a: array of integer;
                      x: integer): integer;
begin
  var k: integer;
  var i:=0;
  var j:=a.Length-1;
  repeat
    k := (i+j) div 2;
    if x>a[k] then
      i := k+1
    else
      j := k-1;
  until (a[k]=x) or (i>j);
  if a[k]=x then
    Result := k
  else
    Result := -1;
end;
```

```
procedure MinElem(a: array of integer;
                  var min, imin: integer);
begin
  imin := 0;
  for var i:=1 to a.Length-1 do
    if a[i]<a[imin] then
      imin := i;
  min := a[imin];
end;
```

1.1 Использование процедурных типов

```
type
  IPredicate = function(x: integer):boolean;

function Even(x: integer): boolean;
begin
  Result := not odd(x);
end;

function IsPositive(x: integer): boolean;
begin
  Result := x>0;
end;

function FindPred(a: array of integer;
            pred: IPredicate): integer;
begin
  var n := a.Length;
  var i := 0;
  while (i<n) and not pred(a[i]) do
    i += 1;
  if i=n then
    Result := -1
  else
    Result := i;
end;

function CountPred(a: array of integer;
              pred: IPredicate): integer;
begin
  Result := 0;
  for var i:=0 to a.Length-1 do
    if pred(a[i]) then
      Result += 1;
end;

procedure MinElemPred(a: array of integer;
  pred: IPredicate; var min, imin: integer);
begin
  min := Integer.MaxValue;
  imin := -1;
  for var i:=0 to a.Length-1 do
    if pred(a[i]) and (a[i]<min) then
    begin
      min := a[i];
      imin := i;
    end;
end;
```

2 Сдвиги, вставка, удаление

2.1 Сдвиги

```
procedure ShiftLeft<T>(a: array of T;
                       ifrom, ito: integer);
begin
  Assert((0 <= ifrom) and (ifrom < ito)
         and (ito < a.Length));
  for var i:=ifrom to ito-1 do
    a[i] := a[i+1];
  a[ito] := default(T);
end;

procedure ShiftLeft<T>(a: array of T);
begin
  ShiftLeft(a, 0, a.Length-1);
end;

procedure ShiftRight<T>(a: array of T;
                        ifrom, ito: integer);
begin
  Assert((0 <= ifrom) and (ifrom < ito)
         and (ito < a.Length));
  for var i:=ito downto ifrom+1 do
    a[i] := a[i-1];
  a[ifrom] := default(T);
end;

procedure ShiftRight<T>(a: array of T);
begin
  ShiftRight(a, 0, a.Length-1);
end;

procedure CycleShiftRight<T>(a: array of T;
                             ifrom, ito: integer);
begin
  Assert((0 <= ifrom) and (ifrom < ito)
         and (ito < a.Length));
  var v := a[ito];
  ShiftRight(a, ifrom, ito);
  a[ifrom] := v;
end;

procedure CycleShiftRight<T>(a: array of T);
begin
  CycleShiftRight(a, 0, a.Length-1);
end;
```

2.2 Вставка элемента

```
procedure Insert<T>(a: array of T;
    var n: integer; value: T; idx: integer);
begin
  Assert((0 <= idx) and (idx < n)
         and (n < a.Length));
  ShiftRight(a, idx, n-1);
  n += 1;
  a[idx] := value;
end;
```

2.3 Удаление элементов

```
procedure Delete<T>(a: array of T;
            var n: integer; idx: integer);
begin
  Assert((0 <= idx) and (idx < n)
         and (n <=a.Length));
  ShiftLeft(a, idx, a.Length-1);
  n-=1;
end;

procedure DeleteAll(a: array of integer;
          var n: integer; pred: IPredicate);
begin
  Assert((0 < n) and (n <= a.Length));
  var j := 0;
  for var i:=0 to n-1 do
    if not pred(a[i]) then
    begin
      a[j] := a[i];
      j += 1;
    end;
  n := j;
end;
```

3 Слияние упорядоченных массивов

```
function Merge(a, b: array of integer;
        n, m: integer): array of integer;
begin
  Assert((0 < n) and (n < a.Length));
  Assert((0 < m) and (m < b.Length));
  a[n] := Integer.MaxValue;
  b[m] := Integer.MaxValue;
  SetLength(Result, n+m);
  var ia := 0;
  var ib := 0;
  for var ir:=0 to n+m-1 do
    if a[ia]<b[ib] then
    begin
      Result[ir] := a[ia];
      ia += 1;
    end
    else
    begin
      Result[ir] := b[ib];
      ib += 1;
    end;
end;
```

4 Сортировки

4.1 Сортировка выбором

```
procedure SortByChoice(a: array of integer);
begin
  for var i:=0 to a.Length-2 do
  begin
    var min := a[i];
    var imin := i;
    for var j:=i+1 to a.Length-1 do
      if a[j]<min then
      begin
        min := a[j];
        imin := j;
      end;
    a[imin] := a[i];
    a[i] := min;
  end;
end;

function IndMinElem(a: array of integer;
              ifrom, ito: integer): integer;
begin
  Assert((0 <= ifrom) and (ifrom < ito)
           and (ito < a.Length));
  Result := ifrom;
  for var i:=ifrom+1 to ito do
    if a[i]<a[Result] then
      Result := i;
end;

procedure Swap<T>(var a, b: T);
begin
  var temp := a;
  a := b;
  b := temp;
end;

procedure SortByChoice2(a: array of integer);
begin
  for var i:=0 to a.Length-2 do
    Swap(a[i],
         a[IndMinElem(a, i, a.Length-1)]);
end;
```

4.2 «Пузырьковая» сортировка

```
procedure BubbleSort(a: array of integer);
begin
  for var i:=0 to a.Length-2 do
    for var j:=a.Length-1 downto i+1 do
      if a[j]<a[j-1] then
        Swap(a[j], a[j-1]);
end;

procedure BubbleSort2(a: array of integer);
begin
  var i := a.Length-1;
  var q: boolean;
  repeat
    q := true;
    for var j:=0 to i-1 do
      if a[j+1]<a[j] then
      begin
        Swap(a[j+1], a[j]);
        q := false;
      end;
    i -= 1;
  until q;
end;
```

4.3 Сортировка вставками

```
procedure SortByInsert(a: array of integer);
begin
  for var i:=1 to a.Length-1 do
  begin
    var x := a[i];
    var j := i-1;
    while (j>=0) and (x<a[j]) do
    begin
      a[j+1] := a[j];
      j -= 1;
    end;
    a[j+1] := x;
  end;
end;
```