1. Вывод
```
procedure WriteArray<T>(a: array of T;
  delim: string := ' ');
begin
  foreach x: T in a do
    write(x,delim);
end;

procedure WritelnArray<T>(a: array of T;
  delim: string := ' ');
begin
  WriteArray(a,delim);
  writeln;
end;
```

2. Заполнение случайными числами
```
function CreateRandomArray(n: integer): array of integer;
begin
  SetLength(Result,n);
  for var i:=0 to n-1 do
    Result[i] := random(100);
end;
```

1-2. Использование стандартного модуля Arrays
```
uses Arrays;
begin
  var a := CreateRandomIntegerArray(10);
  a.Writeln;
  Sort(a);
  a.Writeln(',');
end.
```

3. Инвертирование массива
```
procedure Invert<T>(a: array of T);
begin
  var n := a.Length;
  for var i:=0 to n div 2 - 1 do
    Swap(a[i],a[n-i-1]);
end;
```

4. Поиск
```
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;
```

4a. Поиск с барьером
```
function FindWithBarrier<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;
```

5. Минимальный элемент и его индекс
```
procedure MinElem(a: array of integer;
  var min: integer; var minind: integer);
begin
  min := a[0];
  minind := 0;
  for var i:=1 to a.Length-1 do
    if a[i]<min then
    begin
      min := a[i];
      minind := i;
    end;
end;
```

6. Сдвиг влево
```
procedure ShiftLeft<T>(a: array of T);
begin
  for var i:=0 to a.Length-2 do
    a[i] := a[i+1];
  a[a.Length-1] := default(T);
end;
```

7. Сдвиг вправо
```
procedure ShiftRight<T>(a: array of T);
begin
  for var i:=a.Length-1 downto 1 do
    a[i] := a[i-1];
  a[0] := default(T);
end;
```

8. Циклический сдвиг вправо
```
procedure CycleShiftRight<T>(a: array of T);
begin
  var v := a[a.Length-1];
  for var i:=a.Length-1 downto 1 do
    a[i] := a[i-1];
  a[0] := v;
end;
```

9. Удаление k-того
```
procedure Delete<T>(a: array of T; var n: integer;
  k: integer);
begin
  Assert((0<=k) and (k<n) and (n<=a.Length));
  for var i:=k to n-2 do
    a[i] := a[i+1];
  a[n-1] := default(T);
  n -= 1;
end;
```

10. Вставка на k-тое место
```
procedure Insert<T>(a: array of T; var n: integer;
  k: integer; value: T);
begin
  Assert((0<=k) and (k<=n) and (n<a.Length));
  for var i:=n-1 downto k do
    a[i+1] := a[i];
  a[k] := value;
  n += 1;
end;
```

11. Слияние двух упорядоченных в один упорядоченный
```
// a,b упорядочены по возрастанию
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,m+n);
  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;
```

12. Поиск в упорядоченном массиве
```
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;
```