W pliku binarnym znajdują się informacje o ocenach wszystkich studentów ze wszystkich przedmiotów. Dane te są zapisane w postaci rekordów `TDane` (por. deklaracje poniżej) w dowolnej kolejności. Twoim zadaniem jest utworzenie *indeksu* pliku a następnie wykorzystanie go do wygenerowania posortowanego pliku z ostatecznymi ocenami poszczególnych studentów. Indeks pliku to drzewo BST, w którego węzłach znajduje się numer indeksu oraz lista ocen. Drzewo jest uporzadkowane po numerach indeksów, a każda lista w węzłach — po przedmiotach. Numery indeksów w drzewie nie mogą się powtarzać. Przedmioty w listach też się nie powtarzają. Jeśli w pliku z danymi znajdują się dwie różne oceny z tego samego przedmiotu dotyczące tego samego studenta. to ocena późniejsza anuluje wcześniejszą. Twój program ma składać się z czterech modułów i programu głównego. Program główny ma zapytać użytkownika o nazwę pliku z danymi i pliku wynikowego, utworzyć w pamięci indeks tego pliku, a następnie utworzyć plik wynikowy (posortowany po numerach indeksów, a w obrębie tego samego indeksu po przedmiotach) zawierający końcowe oceny studentów z przedmiotów. Następnie plik wynikowy ma zostać wyświetlony na ekranie rekord po rekordzie (program ma oczekiwać na wciśnięcie klawisza przed wyświetleniem kolejnego rekordu).

A oto interfejsy poszczególnych modułów

```
unit Oceny;
interface
type
  TPrzedmiot = (ALG, ANA, WDM, WDL, EMD, MNU, WDP, TPR, ASD, PAS);
  TOcena = (bdb, dbplus, db, dstplus, dst);
  TIndeks = integer;
  TDane = record
            indeks: TIndeks;
            przedm: TPrzedmiot;
            ocena:  TOcena;
         end;
  procedure WypiszIndeksNaEkran (i: TIndeks);
  procedure WypiszOceneNaEkran (o: TOcena);
  procedure WypiszPrzedmiotNaEkran (p: TPrzedmiot);
  procedure WypiszDaneNaEkran (t: TDane);
  function PorownajIndeksy (i1, i2: TIndeks): integer;
  { jesli indeksy i1 sa rowne wynikiem funkcji jest 0,
    jesli i1 < i2, wynikiem funkcji jest -1, wpp
    wynikiem jest 1 }
  function PorownajPrzedmioty (p1, p2: TIndeks): integer;
  { j. w. ale dla przedmiotow }
end.

unit Listy;
interface
{implementuje listy uporzadkowane rosnaco po polu przedm }
uses
  Oceny;
type
  Lista = ^TElemListy;
  TElemListy = record
                 przedm: TPrzedmiot;
                 ocena:  TOcena;
                 nast: Lista
               end;
procedure Inicjuj (var l: Lista);
{ tworzy pusta liste }
```

```
function CzyPusta (l: Lista): boolean;
{ sprawdza czy l jest lista pusta }
procedure Wstaw (var l: Lista; p: TPrzedmiot; o: TOcena);
{ wstawia przedmiot p z ocena o na wlasciwe miejsce
  w liscie uorzadkowanej l, jesli przedmiot juz
  sie w niej znajdowal to zmienia zwiazana z nim ocene }
function OcenaZ (l: Lista; p: TPrzedmiot; var o: TOcena): boolean;
{ jesli przedmiot p jest na liscie l to wynikiem funkcji jest
  true, a zmienna o to ocena zwiazana z przedmiotem p,
  wpp wynikiem funkcji jest false, a o ma wartosc
  nieokreslona }
procedure UsunZListy (var l: Lista; p: TPrzedmiot);
{ usuwa przedmiot p wraz z ocena z listy l }
procedure Sprzataj (var l : Lista);
{ usuwa wszystkie elementy z listy l, zwalnia pamiec }
end.
```

```
unit BST;
interface
uses
  Oceny, listy;
type
  Drzewo = ^TElemDrzewa;
  TElemDrzewa = record
                  indeks: TIndeks;
                  l : Lista;
                  lewy, prawy: Drzewo
                end;
procedure Inicjuj (var d: Drzewo);
{ Tworzy puste drzewo BST }
procedure Wstaw (var d: Drzewo; i: TIndeks; l: Lista);
{ Wstawia do drzewa d element z numerem indeksu i i lista l,
  jesli element z numerem indeksu i juz jest w drzewie to
  podmienia zwiazana z nim liste (bez dealokacji) }
function Wyszukaj (d: Drzewo; i: TIndeks): Lista;
{ wynikiem jest lista zwiazana z numerem indeksu i lub nil
  jesli i nie ma w drzewie }
procedure Usun (var d: Drzewo; i: TIndeks);
{ usuwa z drzewa (bez dealokacji listy!) wezel
  z numerem indeksu i }
procedure Sprzataj (var d: Drzewo);
{ usuwa wszystkie elementy z drzewa, zwalnia pamiec }
end.
```

```
unit pliki;
interface
type TPlik = file of TDane;
function CzyJest (nazwa: string): boolean;
{ sprawdza, czy plik nazwa istnieje }
function BudujIndeks (var g: TPlik): Drzewo;
{ buduje indeks otwartego do odczytu pliku g }
procedure UtworzPlik (var g: TPlik, d: Drzewo);
{ zapisuje dane do otwartego do zapisu pliku g
  w oparciu o zbudowany indeks d }
```