PLIKI

Plik jest struktura składającą się z elementów tego samego typu.
Liczba elementów składowych pliku jest zmienna. Nie ma zadanych ograniczeń na liczbę elementów pliku, poza rozmiarami pamięci masowych komputera. Długość pliku może być dowolnie zwiększana lub zmniejszana.
Elementy pliku nie posiadają indeksów ani nazw (inaczej niż ma to miejsce w przypadku tablic i rekordów). Elementy pliku ułożone są sekwencyjnie, a program w danej chwili ma dostęp tylko do jednego z nich.
Plik jest logicznym modelem fizycznego zasobu danych. W programie zmienna plikowa kojarzona jest z konkretnym zewnętrznym plikiem zapisanym w pamięci masowej komputera.
Pliki stosowane są w przypadkach, gdy istnieje potrzeba długotrwałego przechowywania danych, wyników pośrednich itp., szczególnie dużych zbiorów informacji na nośnikach masowych.
Program może korzystać z wielu plików jednocześnie.

Składnia deklaracji zmiennej plikowej:

file of <typ>

Występujący w definicji typ oznacza identyfikator dowolnego typu prostego, porządkowego, strukturalnego lub wskaźnikowego, który definiuje typ elementu pliku. Typ elementu nie może być jednak typem plikowym, ani typem złożonym zawierającym elementy typu plikowego.
Jeżeli po słowie kluczowym **file** nie występuje określenie typu elementu, to taki typ służy do deklaracji plików niezdefiniowanych.

Przykłady:

```
type
       liczby = file of integer;
       znaki  = file of char;
```

Typy plikowe *liczby* i *znaki* składają się z elementów będących liczbami całkowitymi lub znakami.

```
type
       mac      =  array [1..10, 1..10] of real;
       wektory  =  file of array [1..10] of real;
       macierze =  file of mac;
```

Identyfikatory *wektory* i *macierze* są nazwami typów plikowych o elementach tablicowych.

```
type
       data     = record
                      dzien:   1..31;
                      miesiac: 1..12;
                      rok:     0..2000;
       student  = record
                      nazwisko, imie: string[20];
                      miejscowosc: string[15];
                      data_ur: data;
                      plec: (mezczyzna, kobieta);
```

```
                semestr: 1..10;
              end;
    daty = file of data;
    studenci = file of student;
```

Elementami typów plikowych *daty* i *studenci* są rekordy.

Najczęściej wykorzystuje się pliki o elementach rekordowych.

W Pascalu istnieje predefiniowany typ plikowy o nazwie TEXT.
Elementami takiego pliku są znaki.
Pliki o deklaracji FILE OF CHAR lub TEXT nazywane są plikami tekstowymi. Mogą być obsługiwane tak jak pliki elementowe (element to jeden znak) lub z wykorzystaniem procedur we-wy.

Jak wynika z zamieszczonego wyżej diagramu składni deklaracji plików możliwe jest używanie plików bez zdefiniowania typu elementu.
Pliki niezdefiniowane nie będą tutaj omawiane.

Dostęp do pliku uzyskuje się za pomocą zmiennej plikowej, tj. zmiennej zadeklarowanej jako typu plikowego.

Przykład:

```
var     numery: liczby;
        symbol: znaki;
        m: macierze;
        grupa: studenci;
```

Wykorzystanie plików w każdym języku programowania wiąże się z przestrzeganiem odpowiedniej procedury postępowania rozpoczynającej się od uzyskania dostępu do fizycznie istniejącego pliku dyskowego pozostającego dotychczas pod kontrola systemu operacyjnego, a kończącą się zwolnieniem go.
W Pascalu obowiązuje następujące następstwo czynności związanych z obsługą plików:

1. Deklaracja zmiennej plikowej
2. Skojarzenie nazwy z fizycznym plikiem zewnętrznym
3. Otwarcie pliku
4. Wykonywanie operacji na pliku (czytanie i pisanie)
5. Zamkniecie pliku

SKOJARZENIE PLIKU Z FIZYCZNYM ZBIOREM DANYCH

Do tego celu służy procedura Assign, wiążąca zewnętrzną nazwę pliku zgodna ze składnią obowiązującą w systemie operacyjnym z daną zmienną plikową.
Wywołanie standardowej procedury ASSIGN ma postać:

```
ASSIGN(nazwa_pliku, string);
```

Pierwszy parametr powinien być zadeklarowaną uprzednio nazwą zmiennej plikowej, a drugi łańcuchem zawierającym zewnętrzną nazwę pliku wraz z ewentualna ścieżką dostępu i rozszerzeniem.

Np.

```
ASSIGN(symbol, 'zbior');
ASSIGN(numery, 'a:\kat\baza.ptp');
```

Struktura elementu kojarzonych fizycznych zbiorów danych powinna być zgodna z definicją typu elementu pliku, który reprezentowany jest przez zmienną.

OTWARCIE PLIKU

Przed rozpoczęciem przetwarzania pliku należy go otworzyć jedną z trzech procedur: Rewrite, Reset lub Append. Sposób wywołania tych procedur zależny jest od rodzaju pliku (zdefiniowany, niezdefiniowany, tekstowy).

Dla wszystkich rodzajów plików dozwolone jest użycie procedur Reset i Rewrite w postaci:

```
Reset (p);
Rewrite (p);
```

gdzie: p - oznacza zmienną plikową.

Wywołanie Rewrite powoduje utworzenie nowego fizycznego pliku danych o nazwie skojarzonej ze zmienna plikowa przy pomocy procedury Assign. Jeżeli plik fizyczny o takiej nazwie istnieje to zostanie skasowany.
W obu przypadkach zostanie utworzony plik pusty, a jego ustawienie będzie wskazywać na początek.

Otwarcie pliku procedura Reset powoduje ustawienie zewnętrznego pliku fizycznego w pozycji początkowej, tj. przed pierwszym elementem. W tym przypadku plik zewnętrzny skojarzony ze zmienną plikową musi istnieć (w przeciwnym przypadku powoduje to powstanie błędu wejścia-wyjścia).

W przypadku plików tekstowych (tj. o definicji FILE OF CHAR lub TEXT) do otwarcia można użyć procedury Append w następujący sposób:

Append (p)

Rezultatem takiego wywołania jest ustalenie pliku na pozycji końcowej umożliwiające dopisywanie tekstów na końcu pliku.

OPERACJE NA PLIKU ELEMENTOWYM (zapis i odczyt z pliku)

Zapis informacji do pliku

Do wprowadzania elementów (z pamięci) do pliku stosuje się procedurę:

```
Write(p, lista_zmiennych)
```

gdzie:
p - oznacza zmienna plikowa,

lista_zmiennych - jest lista zmiennych typu zgodnego z typem elementu pliku, których wartości maja być zapisane pliku zewnętrznego skojarzonego ze zmienna *p*,

Wywołanie procedury standardowej WRITE musi być poprzedzone otwarciem pliku i powoduje wykonanie dla każdej zmiennej z listy_zmiennych następujących dwu czynności:

- zapis do pliku wartości zmiennej,
- przesuniecie pliku o jeden element.

Odczyt informacji z pliku

Standardowa procedura czytania z pliku ma postać:

```
Read(p, lista_zmiennych)
```

gdzie:
p - oznacza zmienna plikowa,
lista_zmiennych - jest lista zmiennych typu zgodnego z typem elementu pliku, którym nadane zostaną wartości kolejnych elementów pobieranych z pliku p,

Wywołanie procedury standardowej READ musi być poprzedzone otwarciem pliku i powoduje wykonanie dla każdej zmiennej z listy_zmiennych następujących dwu czynności:

- odczyt wartości bieżącego elementu z pliku i przypisanie jej pod zmienną,
- przesuniecie pliku o jeden element.

W naszej praktyce dostęp do elementów pliku będziemy uzyskiwali wyłącznie w drodze odczytania ich instrukcja Read. Przyjmujemy, ze elementy pliku nie posiadają swoich identyfikatorów i ich wartości mogą być znane tylko po przepisaniu do zmiennej wymienionej jako drugi (i/lub następny) parametr procedury Read.

ZAMKNIECIE PLIKU

Ostatnią z prezentowanych procedur jest:

```
Close(p)
```

gdzie: p - oznacza zmienna plikowa.

Procedura ta zamyka plik otwarty wcześniej jedna z procedur Reset, Rewrite lub Append. Po tej operacji niedozwolone jest użycie żadnej z funkcji lub procedur z argumentem p oznaczającym zamknięty plik (za wyjątkiem procedur otwarcia i APPEND).
Zamkniecie pliku powoduje wykonanie określonych przez system operacyjny czynności związanych ze zwolnieniem zewnętrznego fizycznego pliku danych. Jednocześnie używana w programie zmienna plikowa zostaje "zwolniona" i może być skojarzona z innym zewnętrznym plikiem.

Przykłady:

1) Napisz program zapisywania w pliku typu FILE OF INTEGER 10 podanych przez użytkownika liczb.

```
Program Liczby;
var buf, i: integer;
    pl: file of integer;
begin
   ASSIGN(pl, 'Liczby');
   REWRITE)pl);
   for i:=1 to 10 do
     begin
     read(buf);
     WRITE(pl, buf);
     end;
   CLOSE(pl);
end.
```

W przykładzie powyższym operacje na pliku prowadzone są w kolejności zgodnej z obowiązującym opisanym wyżej schematem postępowania - deklaracja - skojarzenie z nazwą pliku zawietrznego (Liczby) - operacje zapisu na plik - zamkniecie.
Należy zwrócić uwagę, ze w przykładzie użyto jednak procedury plikowej (READ) w sposób niezgodny ze standardem. Wywołanie w postaci:

read(buf);

nie zostało poprzedzone otwarciem pliku, ani nawet odpowiednią deklaracją. Poza tym w wywołaniu umieszczono tylko jeden parametr - nie ma zmiennej plikowej. Taki sposób operacji plikowych jest dopuszczalny w odniesieniu do pewnych plików predefiniowanych, tu pliku INPUT skojarzonego z klawiaturą. Omówienie takich plików nastąpi poniżej.

2) Napisać program obliczania sumy liczb wpisanych uprzednio do pliku LICZBY.

```
var s: real;  buf, i: integer;
    pl: file of integer;
begin
   ASSIGN(pl, 'Liczby');
   RESET)pl);
   s:=0;
   for i:=1 to 10 do
     begin
     READ(pl, buf);
     s:=s+buf;
     end;
   CLOSE(pl);
   writeln('Suma=',s:8:2);
end.
```

W związku z prezentacja powyższego przykładu należy podkreślić zasadniczą zaletę struktur plikowych: ponieważ pliki są przechowywane w pamięciach zewnętrznych, to po wprowadzeniu danych do pliku można z nich wielokrotnie korzystać, także z innych programów i nawet po długim czasie od chwili zapisu. Ponadto plik stanowi odrębną całość nie związaną na stałe z żadnym programem, a zapisany na przenośnym nośniku może być przeniesiony na inny komputer.

3) Napisać w postaci procedury algorytm zapisywania w pliku danych o N osobach. Dane powinny zawierać nazwisko, wiek i płeć każdej osoby.

```
type osoba = record
               naz:  string[20];
               plec: (kobieta, mezczyzna);
               wiek: byte;
             end;
      plik = file of osoba;

procedure zapisz(var a: plik);
var i,n: integer;
    buf: osoba;
    pom: char;
begin
   readln(n);
   for i:=1 to n do
     begin
     readln(buf.naz);
     readln(pom);
     if (pom='K') or (pom='k') then buf.plec:=kobieta
                               else buf.plec:=mezczyzna;
     readln(buf.wiek);
     WRITE(a, buf);
     end;
end;
```

W przykładzie tym zastosowano plik o elementach rekordowych, składających się z trzech pól. W procedurze umieszczono jedynie wywołanie procedury zapisu na plik WRITE - zakłada się, że pozostałe operacje związane z obsługą pliku przeprowadzone będą w segmencie programu w którym wywołana będzie procedura ZAPISZ.
Jak widać na przykładzie operacje zapisu na plik należy przeprowadzać wpisując od razu cały element. Dlatego w każdym cyklu pętli zapis należy poprzedzić operacja "kompletowania" poszczególnych pól pomocniczej zmiennej rekordowej buf, która pełni role bufora pośredniczącego w przekazywaniu informacji od użytkownika do pliku.
Operacja "kompletowania" jest tu utrudniona ponieważ typ pola płeć jest zdefiniowany przez wyliczenie. Identyfikatory stałych występujące w typie wyliczeniowym: KOBIETA i MEZCZYZNA istnieją tylko w programie i nie mogą być wprowadzone bezpośrednio instrukcją READ.
Informacje o płci osoby użytkownik określa podając literę 'k' dużą lub małą w przypadku kobiety, a dowolny inny znak w przypadku mężczyzny. Dopiero w programie następuje ustalenie odpowiedniej zawartości pola PLEC.
Nie ma natomiast przeszkód do zapisywania wartości wyliczeniowych w polach (czy jako cale elementy) plików elementowych.

Należy zwrócić uwagę, że deklaracja parametru plikowego procedury zawiera słówko VAR - przekazywanie przez zmienną. Z uwagi na fakt, że operacja przypisania zmiennej plikowej nie istnieje nie ma innej możliwości przekazywania plikowych parametrów procedur – przekazywanie przez wartość jest niedopuszczalne.

4) Napisać procedurę drukującą nazwiska i wyznaczającą liczbę kobiet w wieku nie przekraczającym 23 lat, zakładając istnienie pliku o elementach typu OSOBA zdefiniowanego w przykładzie poprzednim.

```
type osoba = record
               naz:  string[20];
               plec: (kobieta, mezczyzna);
               wiek: byte;
```

```
            end;
   plik = file of osoba;

procedure odszukaj(var a: plik; var l: integer);
var i,n: integer;
    buf: osoba;
    pom: char;
begin
   RESET(a);
   l:=0;
   while not EOF(a) do
     begin
     READ(a, buf);
     if (buf.wiek <= 23) and (buf.plec=kobieta) then
       begin
       Writeln('buf.naz);
       l:=l+1;
       end;
     end;
end;
```

W procedurze użyto operacji RESET w celu ustawienia pliku na pierwszym elemencie (by odpowiednio rozpocząć przeglądanie pliku).
Elementy pliku odczytywane są i odpowiednio interpretowane w pętli o nagłówku WHILE NOT EOF(a) DO.
Zastosowano funkcje logiczna EOF sygnalizująca koniec pliku.
Te i niektóre inne pomocniczo stosowane w Pascalu predefiniowane funkcje i procedury opisano poniżej.

Funkcje i procedury operujące na plikach

Podczas odczytu lub zapisu elementów z/do pliku możliwe jest wykorzystanie funkcji i procedur standardowych ułatwiających przetwarzanie pliku:

```
Eof (p)          - pliki elementowe i tekstowe
FilePos (p)      - pliki elementowe i tekstowe
FileSize (p)     - pliki elementowe i tekstowe
Seek (p,pozycja) - pliki elementowe i tekstowe
Truncate (p)     - pliki elementowe i tekstowe
Eoln (p)         - pliki tekstowe
SeekEof (p)      - pliki tekstowe
SeekEoln (p)     - pliki tekstowe
```

Funkcja Eof zwraca wartość **true**, gdy plik znajduje się w pozycji tuż przed znakiem końca pliku lub gdy plik jest pusty. W innych przypadkach Eof daje wartość fal**s**e. Funkcja Eof najczęściej wykorzystywana jest w programie w instrukcjach powtarzania jako warunek dalszego czytania z pliku.

Funkcja FilePos zwraca wartość określającą aktualną pozycję w pliku (liczba LONGINT z przedziału <1,FileSize(p)>). Jeżeli plik jest w pozycji początkowej (np. po wykonaniu procedury Reset lub Rewrite) to funkcja zwraca wartość 0.

Liczbę wszystkich elementów pliku można wyznaczyć przy pomocy funkcji FileSize (wynik - liczba typu LONGINT).

Przetwarzanie plików odbywa się przeważnie w sposób sekwencyjny tzn. element po elemencie. Kolejność tę można zmienić stosując procedurę Seek podając jako drugi argument zadana pozycję pliku (czyli numer elementu). Elementy pliku są numerowane poczynając od 0.

Zastosowanie procedury Truncate powoduje usuniecie z zewnętrznego pliku wszystkich jego elementów począwszy od aktualnej pozycji do końca zbioru.

Procedura Eoln ma zastosowanie w przypadku przetwarzania plików tekstowych i zwraca wartość true, tylko wtedy jeżeli plik znajduje się w pozycji końca wiersza lub końca zbioru.

Funkcje SeekEof lub SeekEoln stosuje się również dla plików tekstowych. Dają podobne wyniki jak Eof i Eoln lecz wykrywają koniec zbioru lub wiersza z ignorowaniem znaków spacji i tabulacji. Funkcje te wykorzystywane są przy wczytywaniu danych numerycznych z plików tekstowych.

Z innych procedur operujących na plikach można wymienić następujące:

Erase (p) - wszystkie rodzaje plików
Rename (p, nowa_nazwa)

Procedura Erase powoduje usuniecie zewnętrznego pliku skojarzonego ze zmienną *p*.

Procedura Rename umożliwia zmianę nazwy zbioru na nową wyspecyfikowaną w parametrze *nowa_nazwa* typu łańcuchowego. Procedura ta nie może być zastosowana w przypadku pliku otwartego.

Pliki tekstowe

Ogólna definicja procedur WRITE i READ:

read(plik, lista_zmiennych);
readln(plik, lista_zmiennych);
write(plik, lista_wyrazen);
writeln(plik, lista_wyrazen);

Pierwszy parametr tych procedur to nazwa pliku. Parametr ten może być pominięty. Jeśli tak jest to Pascal przyjmuje, że czytanie i wydruk odbywa się z domyślnych plików INPUT i OUTPUT (standardowo ustawione są klawiatura i ekran).
Jeśli parametr pierwszy jest wymieniony, to powinien być zmienną plikową skojarzoną z otwartym plikiem. Operacje we-wy dotyczą tego pliku.
Jeśli jest to plik tekstowy to możliwe jest czytanie liczb, znaków i napisów oraz wydruk formatowany jak w przypadku standardowej obsługi we-wy.

Przykład

```
procedure ZapiszDane(var plik: text; n: integer);
var i: integer;
    x, y: real;
begin
writeln('Wprowadz ', n, ' wierszy. Po dwie liczby w wierszu');
```

```
rewrite(plik);
for i:=1 to n do
  begin
  writeln(i, ' wiersz');
  write('Pierwsza liczba: '); readln(x);
  write('Druga liczba: '); readln(y);
  writeln(plik, x, y);
//  writeln(plik, x:8:2, y);
  end;
close(plik);
end;
```

```
procedure WyswietlDane(var plik: text);
var i: integer;
   x, y: real;
begin
writeln('Wydruk zawartosci pliku');writeln;
reset(plik);
i:=0;
while not eof(plik) do
 begin
 i:=i+1;
 readln(plik, x, y);
 writeln(i, ' wiersz:', x, '  ',y);
 end;
writeln;writeln('Wyprowadzono ', i, ' linii');
close(plik);
end;
```

```
procedure WyswietlDane2(var plik: text);
var i: integer;
   x: real;
begin
writeln('Wydruk zawartosci pliku');writeln;
reset(plik);
i:=0;
while not eof(plik) do
 begin
 i:=i+1;
 write(i, ' wiersz:');
 while not eoln(plik) do
   begin
   read(plik, x);
   write(x, '  ');
   end;
 readln(plik);
 writeln;
 end;
writeln;writeln('Wyprowadzono ', i, ' linii');
close(plik);
end;
```

```
var p: text;

begin
assign(p, 'dane.txt');
ZapiszDane(p, 5);
WyswietlDane(p);
WyswietlDane2(p);
readln;
end.
```