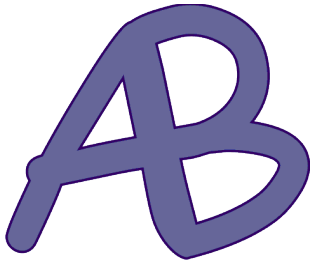

Wykład
PASCAL
- Pliki tekstowe

Rodzaje plików

Dane przechowywane w pliku mogą mieć reprezentację binarną (taką samą, jak w pamięci komputera) lub tekstową (taką, jaka używana jest do wprowadzania informacji z klawiatury i wyprowadzania jej na ekran monitora lub drukarkę).

Reprezentacjom tym odpowiadają w Pascalu pliki:

- ✓ **elementowe** (inaczej nazywane zdefiniowanymi lub binarnymi) – mogą przechowywać dane dowolnego typu. Ich interpretacja zależy od programu, który je odczytuje. Wszystkie dane przechowywane w plikach elementowych muszą być tego samego typu.
- ✓ **tekstowe** – przechowują tekst w zapisany w kodzie ASCI

Rodzaje plików

Zawartość plików elementowych jest na ogół nieczytelna dla użytkownika. Treść pliku tekstowego daje można odczytać w każdym programie obsługującym kod ASCI.

Pliki tekstowe mogą być użyte do przechowywania mieszanych typów danych (np. tekstów i liczb), gdyż wszelka informacja przechowywana jest w nich w postaci kolejnych linii z zawartością.

Pliki elementowe są plikami o dostępie swobodnym, to znaczy, że w każdym momencie można odwołać się do dowolnego elementu pliku.

Pliki tekstowe są plikami o dostępie sekwencyjnym, co oznacza, że aby dostać się do wybranego elementu pliku, należy przeczytać wszystkie elementy znajdujące się przed nim.

Pliki tekstowe

Pliki tekstowe

- ✓ Każdy bajt interpretowany jest jako znak w kodzie ASCII.
- ✓ Najmniejszym elementem jaki można odczytać jest linia, czyli ciąg znaków ograniczony znakiem końca linii.

Aby móc używać pliku deklaruje się tzw. zmienną plikową, w której przechowywany będzie uchwyt do pliku.

```
var
   nazwa_zmiennej_plikowej : text;
```

Np.:

```
var
    plik:Text;
```

Pliki tekstowe

Operacja na pliku w Pascalu przebiega w czterech etapach:

1. Przypisanie zmiennej plikowej do pliku na dysku
2. Otwarcie lub utworzenie i otwarcie pliku powiązanego przez zmienną.
3. Operacje na danych (zapis lub odczyt)
4. Zamknięcie pliku

Otwarcie pliku

Plik można otworzyć na trzy sposoby:

1. Reset (zmienna_plikowa) - umożliwia otwarcie już istniejącego pliku, ustawiając tzw. wskaźnik plikowy na jego początku. W przypadku, gdy otwierany plik nie istnieje, wywołanie procedury reset zakończy się błędem wykonania.

2. Rewrite (zmienna_plikowa) - umożliwia otwarcie pliku niezależnie od tego, czy istniał on poprzednio: jeśli plik nie istnieje utworzony zostanie nowy plik o danej nazwie, zaś jeśli plik istniał - zeruje długość istniejącego pliku i ustawia wskaźnik plikowy na jego początku (czego efektem jest utracenie wszystkich danych zawartych w pliku)

Otwarcie pliku

3. Append (zmienna_plikowa) - otwiera plik do dopisywania, tj. otwiera go do zapisu nie niszcząc poprzedniej zawartości i ustawia wskaźnik plikowy na jego końcu. Umożliwia to dodawanie danych do plików tekstowych

Należy pamiętać, że w przypadku plików tekstowych procedura reset otwiera plik wyłącznie do odczytu, zaś rewrite - wyłącznie do zapisu (nie ma zatem możliwości mieszania odczytów i zapisów w jednym cyklu otwarcia).

Obsługa błędów

Podczas pracy z plikami, może się zdarzyć, że otwierany plik nie istnieje lub nie ma do niego dostępu. W takiej sytuacji program zostanie przerwany.

Pascal udostępnia dyrektywy kompilatora, które pozwalają na przejęcie kontroli nad pojawiającymi się błędami. Taką dyrektywą jest **{$I-}** oraz **{$I+}** Gdy pomiędzy nimi umieścimy kod, będzie on odporny na błędy wejścia/wyjścia, inaczej mówiąc odporny również na błędy dotyczące obsługi plików,

Obsługa błędów

Funkcja sprawdzająca czy plik istnieje:

```
1  function CzyPlikIstnieje(const NazwaPliku : string) : Boolean;
2  var F : File;
3  begin
4    {$I-}
5      Assign(F, Nazwa Pliku);
6      Reset(F);
7      Close(F);
8    {$I+}
9
10   CzyPlikIstnieje := IOResult = 0 ;
11 end;
```

Rezultat wykonania operacji odczytujemy funkcją IOResult. Jeżeli zwróci ona wartość równą 0 oznacza to, że wykonanie operacji we-wy przebiegło pomyślnie. Każdej innej wartości przyporządkowany jest określony typ błędu.

Obsługa błędów

Funkcja sprawdzająca czy plik istnieje, jeżeli tak – otwiera go do odczytu, w przeciwnym razie tworzy nowy plik.

```
1   function otworzplik (nazwapliku : string) : Boolean;
2
3   begin
4   assign (plik,nazwapliku);
5   {$I-}
6   reset (plik);
7   if ioresult<>0 then rewrite(plik);
8   if ioresult=0 then otworzplik:=False else
9   otworzplik:=True;
10  {$I+}
11  end;
```

Operacje zapisu i odczytu danych

Do wymiany danych pomiędzy programem a plikiem służą procedury read i write (zapis).
Ponieważ w standardowej wersji obsługują one ekran monitora i klawiaturę, niezbędne jest podanie dodatkowego argumentu określającego plik,(zmiennej plikowej).

```
read(zmienna_plikowa, lista_elementów);

write(zmienna_plikowa, lista_elementów);
```

Dla plików tekstowy możliwe jest użycie procedur readln i writeln, odczytujących lub zapisujących dane wraz ze znakami końca wiersza.

Operacje zapisu i odczytu danych

- Write(Plik, v1, v2, ..., vn) - procedura ta zapisuje do pliku zewnętrznego skojarzonego ze zmienną plikową **Plik** zmienne **v1, v2, ..., vn** (muszą być one tego samego typu co typ podstawowy zmiennej plikowej). Po zapisie każdej, kolejnej pozycji wskaźnik pliku jest przesuwany do następnej składowej pliku. Jeżeli wskaźnik jest na końcu pliku, plik jest rozszerzony (nowe informacje są dopisywane do pliku).

- Read(Plik, v1, v2, ..., vn) - procedura ta czyta z pliku zewnętrznego związanego ze zmienną plikową **Plik** zmienne **v1, v2, ..., vn.**. Przy czytaniu każdej zmiennej wskaźnik pliku jest przesuwany do następnej składowej pliku. Gdy wskaźnik pliku znajduje się na końcu pliku, próba czytania z pliku spowoduje błąd wejścia/wyjścia.

Operacje zapisu i odczytu danych

- Eof(Plik) - funkcja ta podaje czy osiągnięto koniec pliku, tzn. **Eof(Plik)=TRUE** jeżeli osiągnięto koniec pliku.

- Close(Plik) - procedura ta zamyka plik dyskowy związany ze zmienną plikową **Plik**, który został uprzednio otwarty przy użyciu procedur **Reset** lub **Rewrite**. Plik jest uaktualniany. Wykonuj tę procedurę obowiązkowo, jeśli nie chcesz już używać pliku.

- Erase(Plik) - procedura ta kasuje plik skojarzony ze zmienną plikową **Plik**. Nie jest dopuszczalne stosowanie Erase w stosunku do otwartego pliku. Plik musi być zamknięty procedurą **Close** przed użyciem procedury Erase.

Operacje zapisu i odczytu danych

- Eof(Plik) - funkcja ta podaje czy osiągnięto koniec pliku, tzn. **Eof(Plik)=TRUE** jeżeli osiągnięto koniec pliku.

- Close(Plik) - procedura ta zamyka plik dyskowy związany ze zmienną plikową **Plik**, który został uprzednio otwarty przy użyciu procedur **Reset** lub **Rewrite**. Plik jest uaktualniany. Wykonuj tę procedurę obowiązkowo, jeśli nie chcesz już używać pliku.

- Erase(Plik) - procedura ta kasuje plik skojarzony ze zmienną plikową **Plik**. Nie jest dopuszczalne stosowanie Erase w stosunku do otwartego pliku. Plik musi być zamknięty procedurą **Close** przed użyciem procedury Erase.

Operacje zapisu i odczytu danych

```
1   program zapis_i_odczyt_z_pliku;
2   var F:Text;
3       s:string;
4   begin
5    //zapis do pliku
6    Assign(F, 'plik.txt');
7    Rewrite(F);
8    WriteLn(F, 'Ala ma kota');
9    Close(F);
10
11   //odczyt z pliku
12   Assign(F, 'plik.txt');
13   Reset(F);
14   ReadLn(F, s);
15   Close(F);
16   WriteLn(s);
17  end.
```

Operacje zapisu i odczytu danych

Program czyta zawartość pliku DANE.TXT

```
1  program czytaj_plik;
2  uses crt;
3  var
4    plik : text;
5    wiersz : string;
6  begin
7    clrscr;
8    Writeln('Zawartosc pliku dane.txt jest nastepujaca: ');
9    writeln('');;
10   assign(plik,'dane.txt');
11   reset(pt);
12   repeat  { Czytanie i wypisywanie kolejnych wierszy az do konca pliku}
13     readln(plik,wiersz);
14     writeln(wiersz,' ');
15   until eof(plik);
16   close(plik);  { Zamykanie pliku }
17   writeln('');
18   write('Koniec pliku');
19   repeat until keypressed
20 end.
```

Literatura:

W prezentacji wykorzystano przykłady i fragmenty:

- R. Jarża, *Turbo Pascal. Szkoła programowania*, Wydawnictwo Robomatic 2000. *(dostępne w bibliotece uczelni)*
- http://pascal.kurs-programowania.pl
- http://4programmers.net
- T. M. Sadowski, *Turbo Pascal. Programowanie*, Helion 1996.