Лабораторна робота № 8

Використання файлів для зберігання інформації

Мета роботи: ознайомитись з принципами застосування та практично засвоїти використання зовнішніх файлів у програмах на TurboPascal (FreePascal).

Теоретичні відомості

Програми, що розроблялись раніше в якості пристрою вводу-виводу використовували лише системну консоль – клавіатуру та екран монітора. У ряді випадків з метою вводу – виводу доцільніше використовувати зовнішні файли, розміщені на дискових накопичувачах. Особливо файли виявляються корисними, коли програма оперує великою кількістю вхідної і (або) вихідної інформації. Корисними файли є і з огляду на сучасні тенденції та розвиток електронних засобів зв'язку – файл з даними чи результатами простіше пересилати електронною поштою.

Файл, з погляду програми мовою Pascal, — це іменована структура даних, що представляє собою послідовність елементів одного або декількох типів. Кількість елементів файлу практично не обмежена. У першому наближенні файл можна розглядати як масив змінної довжини, розміщений на зовнішньому носії.

Файл – це один з стандартних типів даних Turbo Pascal, тому його можна використовувати як при описі типів, так і при описі змінних. Як і будь-яка змінна у програмі, файл повинен бути описаний у розділі опису змінних. При цьому вказується тип елементів файлу. У загальному вигляді опис файлу виглядає так:

```
FName:file of ElementsType;
```

Наприклад:

```
Res:file of char; { файл символів }
Koef:file of real; { файл дійсних чисел }
f:file of integer; { файл цілих чисел }
```

Такі файли можуть використовуватись для запису – зберігання – зчитування даних лише одного (вказаного типу) в двоїчній формі запису.

Окремим випадком є використання файлів без типу. Такий файл описується як text:

```
res: text.
```

Такий файл може містити дані будь-якого типу у вигляді близькому до їх друкованого формату. Файли типу text зручно застосовувати для запису – зберігання – зчитування різнотипних даних заданої структури. Наприклад, у файл може бути послідовно записана інформація: рядкова змінна – символьний код досліду, послідовність чисел – умови проведення досліду, ціле число – кількість експериментальних точок, масив комплектів чисел – набір експериментальних даних.

З іншого боку, якщо такий файл містить тільки безіменні числа, то необхідно докладно документувати методику його створення.

Файли типу text можна використовувати і як електронні аналоги друкуючого пристрою чи просто перенаправляти в них вивід з монітора.

Опис змінної типу **file** чи **text** ще не створює файла, а лише ініціює роботу служби файлового вводу – виводу. Щоб програма могла взаємодіяти з файлом, файлову змінну необхідно зв’язати з конкретним файлом, що існує на зовнішньому носієві, або що буде там створюватись. Для виконання цієї операції використовується процедура **ASSIGN**, яка в загальному випадку має вигляд:

```
assign(var f:text; FileName:string);
```

Ім'я файлу задається відповідно до прийнятих в MS-DOS правил. Воно може бути повним, тобто складатися не тільки безпосередньо з імені файлу, але і включати шлях до файлу.

Нижче наведено приклади виклику процедури ASSIGN:

```
assign (t,'a:\result.txt');
assign(f,'\students\ivanov\korni.txt') ;
fname:='otchet.txt';
```

```
assign(f,fname) ;
```

Безпосередньо виведення в текстовий файл здійснюється за допомогою інструкцій **WRITE** чи **WRITELN**, у яких список виведених значень починається зі змінної типу text, що ідентифікує файл для виведення. Наприклад, якщо змінна **f** має тип text, то інструкція виведення у файл може бути такою:

```
Write (f, 'Корені рівняння',x1,x2) ;
```

Однак, для того, щоб інструкція **WRITE** вивела дані у файл, не досить призначити ім'я файловій змінній. Потрібно ще відкрити файл для виведення.

Якщо програма, що формує вихідний файл, вже використовувалася, то можливо, що файл з результатами роботи програми вже є на диску. Тому потрібно вирішити, що зробити зі старим файлом: замінити старі дані новими чи додати нові дані до старих. Це визначається під час відкриття файлу.

Можливі наступні варіанти відкриття файлу для запису в нього даних:

- перезапис (запис нового файлу поверх існуючого чи створення нового файлу);
- додавання в існуючий файл.

Щоб відкрити файл у режимі створення нового файлу чи заміни існуючого, необхідно викликати процедуру **REWRITE(f)**, де **f** — файлова змінна типу text або **FILE**.

Щоб відкрити файл у режимі додавання в існуючий файл, необхідно викликати процедуру **APPEND(f)**, де **f** — файлова змінна типу text або **FILE**.

Наведемо приклад програми, що відкриває файл у режимі створення нового файлу і записує в нього 5 рядків.

```
var
  f:text;( текстовий файл }
  i:integer;
Begin
  assign(f,'test.txt');
  rewrite(f) ;    { відкрити в режимі перезапису }
  for i:=1 to 3 do
  writein(f,'Рядок ',i);
```

```
  close(f);        { закрити файл }
End.
```

В результаті виконання програми на диску з'являється файл **test.txt**, в якому буде знаходитись така інформація:

```
Рядок 1
Рядок 2
Рядок 3
```

Наступна програма відкриває створений попередньою програмою файл і записує в нього два рядки.

```
var
  f:text;     { текстовий файл }
  i:integer;
Begin
  assign(f,'test.txt') ;
  rewrite(f);           { відкрити в режимі перезапису }
  for i:=4 to 5 do
  writein(f,'Рядок ',i);
  close(f);                { закрити файл }
End.
```

В результаті виконання програми файл **test.txt** змінюється – у ньому буде міститися інформація:

```
Рядок 4
Рядок 5
```

Якщо у останній програмі оператор **rewrite(f)** замінити оператором **append(f)**, то інформація буде додаватись до уже існуючої і у файлі буде міститися інформація:

```
Рядок 1
Рядок 2
Рядок 3
Рядок 4
Рядок 5
```

Спроба відкрити файл може завершитися невдачею і викликати помилку часу виконання програми. Причин може бути декілька. Наприклад, якщо програма намагається відкрити файл на гнучкому диску, що не готовий

до роботи. Інша причина — відсутність файлу, що відкривається в режимі додавання.

З'ясувати, чи завершилася успіхом процедура відкриття файлу, можна, перевіривши значення функції **`IOResult`** (Input-Output Result — результат вводу-виводу). Функція **`IOResult`** повертає 0, якщо операція вводу-виводу завершилася успішно; в іншому випадку — не нуль. Однак, щоб програма могла перевірити результат виконання операції вводу-виводу, потрібно дозволити їй це зробити, додавши перед викликом процедури відкриття файлу рядок {$I-}. Після інструкції відкриття файлу варто помістити {$I+}. На рис. 8.1 наведено блок-схему алгоритму, а нижче фрагмент програми, що забезпечує відкриття файлу для додавання, якщо файл уже є на диску, чи створення нового, якщо файлу немає.

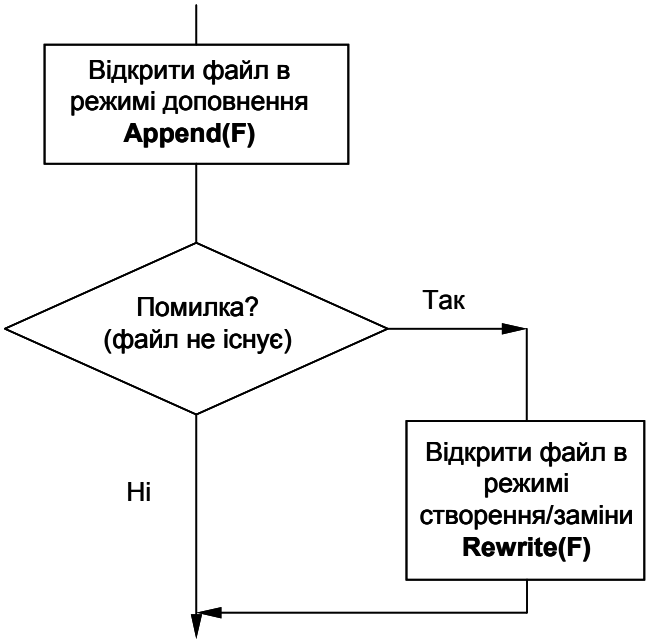

Рисунок 8.1. Блок-схема алгоритму відкриття файлу з обробкою можливої помилки.

```
assign(f,filename);
{$I-}
```

```
append(f);
{$I+}
if IOResult<>0
then rewrite(f);
```

Звичайно, наведена програма є найпростішою і не враховує усю різноманітність причин, які можуть викликати помилку відкриття файлу. Для ознайомлення з повним переліком значень, які повертає **IOResult**, можна скористатися підсистемою допомоги інтегрованого середовища **Turbo Pascal**.

Перед завершенням роботи програма повинна закрити усі відкриті файли. Для цього викликають процедуру **CLOSE**. Процедура **CLOSE** має один параметр – ім'я файлової змінної. Приклад використання процедури: **close (f)**.

В TurboPascal файли можна використовувати і для введення інформації. Щоб скористатися цією можливістю, необхідно виконати описані вище дії (опис змінної типу **text** або **FILE OF** та зв’язування його з файлом на фізичному носієві), відкрити файл для читання і прочитати дані, використовуючи інструкцію **READ** чи **READLN**.

Активація файлу для вводу з нього виконується за допомогою процедури **RESET**, що має один параметр — файлову змінну. Перед викликом процедури **RESET** файловій змінній призначається ім'я файлу викликом процедури **ASSIGN**.

Наступні інструкції відкривають файл для вводу:

```
assign(f,'c:\data.txt');
reset(f);
```

Якщо ім'я файлу зазначене невірно, тобто файлу з таким ім'ям на диску не існує, то виникає помилка часу виконання програми. Слід зазначити, що, як при активації файлу для виводу, може існувати ряд різних причин, що не

дозволять виконати процедуру. І в цьому випадку можна для перехоплення та обробки помилки можна скористатися функцією **IOResult**

Наведемо приклад програми, що використовує значення функції **IOResult** для перевірки відкриття файлу. Якщо спроба відкрити файл викликає помилку, то виникає запит про необхідність повторного відкриття.

Зчитування з файлу виконується за допомогою інструкцій **READ** або **READLN** (тільки у випадку використання файла без типу), що у загальному вигляді записуються так:

```
read(FileName, VarList);
readln(FileName, VarList);
```

де **FileName** — змінна типу **text** або **FILE** (тільки для оператора **Read**), **VarList** — імена змінних, розділені комами.

Процедура **RESET** не тільки (і не обов'язково) готує файл до вводу з нього інформації. Фактично ця процедура виставляє спеціальну мітку в самий початок файла. Після кожної операції зчитування, така мітка пересувається до наступного елемента. При цьому система дозволяє проводити не тільки читання, але й запис інформації. Однак, необхідно враховувати, що запис буде проводитись в поточному місці файлу, і може пошкодити уже наявну там інформацію.

Оскільки файли типу **text** можуть містити різну інформацію, яка може займати різні об'єми, то щодо таких файлів запис в середину файла не рекомендується. Типізовані файли характерні тим, що кожен записаний в них елемент займає однаковий, властивий типу даних об'єм, тому, запис з вільним доступом замінить одні дані іншими.

Для більш гнучкого застосування файлових змінних в **Turbo Pascal** існує ще ряд процедур та функцій, основні з яких наведено нижче.

Функція **EOF** (End Of File) – повертає булеве значення, яке свідчить про досягнення кінця файлу.

Функція **FilePos** – повертає значення типу **LongInt**, яке вказує поточне положення мітки всередині файлу в одиницях типу.

Процедура **SEEK** – дозволяє перемістити мітку відразу у задану позицію.

Процедура **TRUNCATE** – дозволяє відкинути хвіст файла, починаючи з поточної позиції мітки, виставляє в даній позиції мітку кінця файла.

Нижче наведено приклад застосування функції **EOF**, а з рештою процедур та функцій можна познайомитись в додатковій літературі [2], [3], а також в системі електронної підказки середовища Turbo Pascal.

```
Function EOF(var f:text):boolean;
```

У функції **EOF** один параметр - файлова змінна. Функція **EOF** повертає **TRUE**, якщо досягнуто кінця файлу, інакше - **FALSE**.

```
var
  f:text;     { файлова змінна }
  incom:real;{ доход-значення з чергового рядка файлу}
  sumin: real;    { сумарний доход }
Begin
assign(f,'incom.txt');
     {$!-} reset(f); {відкриємо для читання} {$!+}
     if IOResult=0 then
     begin
       sumin:=0;
       writeln('Читання даних.');
       while NOT EOF(f) do {поки не досягнений кінець
  файлу}
       begin
          readln(f,incom);
          sumin: =sumin + incom;
       end;
       close(f);
       writein ('Сумарний доход: ', sumin: 11:2) ;
     end;
     else writeln('Помилка відкриття файлу');
     end.
```

Потрібно також звернути увагу на те, що значення **EOF** перевіряється перед кожним зчитуванням, у тому числі і перед першим. Перед першим зчитуванням **EOF** перевіряється, тому що файл, який відкривається, може існувати на диску, але він може бути порожнім (не містити рядків).

Завдання

Розробити, скласти та відлагодити програму за вказівкою викладача з використанням файлових даних.

Варіанти:

- програми розв'язання системи лінійних рівнянь з читанням даних з текстового файла;
- програми роботи з даними типу запис з використанням файлів типу запис;
- робота з прямим доступом до даних типізованого файлу.

Зробити висновки.

Рекомендована література: [3], [5], [9].