PASCAL - FILE HANDLING

Pascal treats a file as a sequence of components, which must be of uniform type. A file's type is determined by the type of the components. File data type is defined as −

```
type
file-name = file of base-type;
```

Where, the base-type indicates the type of the components of the file. The base type could be anything like, integer, real, Boolean, enumerated, subrange, record, arrays and sets except another file type. Variables of a file type are created using the *var* declaration −

```
var
f1, f2,...: file-name;
```

Following are some examples of defining some file types and file variables −

```
type
   rfile = file of real;
   ifile = file of integer;
   bfile = file of boolean;
   datafile = file of record
   arrfile = file of array[1..4] of integer;

var
   marks: arrfile;
   studentdata: datafile;
   rainfalldata: rfile;
   tempdata: ifile;
   choices: bfile;
```

Creating and Writing to a File

Let us write a program that would create a data file for students' records. It would create a file named students.dat and write a student's data into it −

```
program DataFiles;
type
   StudentRecord = Record
      s_name: String;
      s_addr: String;
      s_batchcode: String;
   end;

var
   Student: StudentRecord;
   f: file of StudentRecord;

begin
   Assign(f,'students.dat');
   Rewrite(f);
   Student.s_name := 'John Smith';
   Student.s_addr := 'United States of America';
   Student.s_batchcode := 'Computer Science';
   Write(f,Student);
   Close(f);
end.
```

When compiled and run, the program would create a file named *students.dat* into the working directory. You can open the file using a text editor, like notepad, to look at John Smith's data.

Reading from a File

We have just created and written into a file named students.dat. Now, let us write a program that would read the student's data from the file −

```
program DataFiles;
type
   StudentRecord = Record
      s_name: String;
      s_addr: String;
      s_batchcode: String;
   end;

var
   Student: StudentRecord;
   f: file of StudentRecord;

begin
   assign(f, 'students.dat');
   reset(f);
   while not eof(f) do

   begin
      read(f,Student);
      writeln('Name: ',Student.s_name);
      writeln('Address: ',Student.s_addr);
      writeln('Batch Code: ', Student.s_batchcode);
   end;

   close(f);
end.
```

When the above code is compiled and executed, it produces the following result −

```
Name: John Smith
Address: United States of America
Batch Code: Computer Science
```

Files as Subprogram Parameter

Pascal allows file variables to be used as parameters in standard and user-defined subprograms. The following example illustrates this concept. The program creates a file named rainfall.txt and stores some rainfall data. Next, it opens the file, reads the data and computes the average rainfall.

Please note that, **if you use a file parameter with subprograms, it must be declared as a var parameter.**

```
program addFiledata;
const
   MAX = 4;
type
   raindata = file of real;

var
   rainfile: raindata;
   filename: string;
procedure writedata(var f: raindata);

var
   data: real;
   i: integer;

begin
   rewrite(f, sizeof(data));
   for i:=1 to MAX do

   begin
      writeln('Enter rainfall data: ');
      readln(data);
      write(f, data);
   end;

   close(f);
end;

procedure computeAverage(var x: raindata);
var
   d, sum: real;
   average: real;

begin
   reset(x);
   sum:= 0.0;
   while not eof(x) do

   begin
      read(x, d);
      sum := sum + d;
   end;

   average := sum/MAX;
   close(x);
   writeln('Average Rainfall: ', average:7:2);
end;

begin
   writeln('Enter the File Name: ');
   readln(filename);
   assign(rainfile, filename);
   writedata(rainfile);
   computeAverage(rainfile);
end.
```

When the above code is compiled and executed, it produces the following result −

```
Enter the File Name:
rainfall.txt
Enter rainfall data:
34
Enter rainfall data:
45
Enter rainfall data:
56
Enter rainfall data:
78
Average Rainfall: 53.25
```

Text Files

A text file, in Pascal, consists of lines of characters where each line is terminated with an end-of-line marker. You can declare and define such files as −

```
type
file-name = text;
```

Difference between a normal file of characters and a text file is that a text file is divided into lines, each terminated by a special end-of-line marker, automatically inserted by the system. The following example creates and writes into a text file named contact.txt −

```
program exText;
var
   filename, data: string;
   myfile: text;

begin
```

```
   writeln('Enter the file name: ');
   readln(filename);

   assign(myfile, filename);
   rewrite(myfile);

   writeln(myfile, 'Note to Students: ');
   writeln(myfile, 'For details information on Pascal Programming');
   writeln(myfile, 'Contact: Tutorials Point');
   writeln('Completed writing');

   close(myfile);
end.
```

When the above code is compiled and executed, it produces the following result −

```
Enter the file name:
contact.txt
Completed writing
```

Appending to a File

Appending to a file means writing to an existing file that already has some data without overwriting the file. The following program illustrates this −

```
program exAppendfile;
var
   myfile: text;
   info: string;

begin
   assign(myfile, 'contact.txt');
   append(myfile);

   writeln('Contact Details');
   writeln('webmaster@tutorialspoint.com');
   close(myfile);

   (* let us read from this file *)
   assign(myfile, 'contact.txt');
   reset(myfile);
   while not eof(myfile) do

   begin
      readln(myfile, info);
      writeln(info);
   end;
   close(myfile);
end.
```

When the above code is compiled and executed, it produces the following result −

```
Contact Details
webmaster@tutorialspoint.com
Note to Students:
For details information on Pascal Programming
Contact: Tutorials Point
```

File Handling Functions

Free Pascal provides the following functions/procedures for file handling −

Sr.No.	Function Name & Description
1	

procedure Append*vart*: *Text***;**

Opens a file in append mode

2

procedure Assign*outf*: *file*; *constName*: **;**

Assigns a name to a file

3

procedure Assign*outf*: *file*; *p*: *PChar***;**

Assigns a name to a file

4

procedure Assign*outf*: *file*; *c*: *Char***;**

Assigns a name to a file

5

procedure Assign*outf*: *TypedFile*; *constName*: **;**

Assigns a name to a file

6

procedure Assign*outf*: *TypedFile*; *p*: *PChar***;**

Assigns a name to a file

7

procedure Assign*outf*: *TypedFile*; *c*: *Char***;**

Assigns a name to a file

8

procedure Assign*outt*: *Text*; *consts*: **;**

Assigns a name to a file

9

procedure Assign*outt*: *Text*; *p*: *PChar***;**

Assigns a name to a file

10

procedure Assign*outt*: *Text*; *c*: *Char***;**

Assigns a name to a file

11

procedure BlockRead*varf*: *file*; *varBuf*; *count*: *Int*64; *varResult*: *Int*64**;**

Reads data from a file into memory

12

procedure BlockRead*varf*: *file*; *varBuf*; *count*: *LongInt*; *varResult*: *LongInt***;**

Reads data from a file into memory

13

procedure BlockRead*varf*: *file*; *varBuf*; *count*: *Cardinal*; *varResult*: *Cardinal*;

Reads data from a file into memory

14

procedure BlockRead*varf*: *file*; *varBuf*; *count*: *Word*; *varResult*: *Word*;

Reads data from a file into memory

15

procedure BlockRead*varf*: *file*; *varBuf*; *count*: *Word*; *varResult*: *Integer*;

Reads data from a file into memory

16

procedure BlockRead*varf*: *file*; *varBuf*; *count*: *Int*64;

Reads data from a file into memory

17

procedure BlockWrite*varf*: *file*; *constBuf*; *Count*: *Int*64; *varResult*: *Int*64;

Writes data from memory to a file

18

procedure BlockWrite*varf*: *file*; *constBuf*; *Count*: *LongInt*; *varResult*: *LongInt*;

Writes data from memory to a file

19

procedure BlockWrite*varf*: *file*; *constBuf*; *Count*: *Cardinal*; *varResult*: *Cardinal*;

Writes data from memory to a file

20

procedure BlockWrite*varf*: *file*; *constBuf*; *Count*: *Word*; *varResult*: *Word*;

Writes data from memory to a file

21

procedure BlockWrite*varf*: *file*; *constBuf*; *Count*: *Word*; *varResult*: *Integer*;

Writes data from memory to a file

22

procedure BlockWrite*varf*: *file*; *constBuf*; *Count*: *LongInt*;

Writes data from memory to a file

23

procedure Close*varf*: *file*;

Closes a file

24

procedure Close*vart*: *Text*;

Closes a file

25	**function EOF**$varf: file$**:Boolean;**	Checks for end of file
26	**function EOF**$vart: Text$**:Boolean;**	Checks for end of file
27	**function EOF: Boolean;**	Checks for end of file
28	**function EOLn**$vart: Text$**:Boolean;**	Checks for end of line
29	**function EOLn: Boolean;**	Checks for end of line
30	**procedure Erase**$varf: file$**;**	Deletes file from disk
31	**procedure Erase**$vart: Text$**;**	Deletes file from disk
32	**function FilePos**$varf: file$**:Int64;**	Position in file
33	**function FileSize**$varf: file$**:Int64;**	Size of file
34	**procedure Flush**$vart: Text$**;**	Writes file buffers to disk
35	**function IOResult: Word;**	Returns result of last file IO operation
36	**procedure Read**$varF: Text; Args: Arguments$**;**	Reads from file into variable

37

procedure Read*Args*: *Arguments*;

Reads from file into variable

38

procedure ReadLn*varF*: *Text*; *Args*: *Arguments*;

Reads from file into variable and goto next line

39

procedure ReadLn*Args*: *Arguments*;

Reads from file into variable and goto next line

40

procedure Rename*varf*: *file*; *consts*: ;

Renames file on disk

41

procedure Rename*varf*: *file*; *p*: *PChar*;

Renames file on disk

42

procedure Rename*varf*: *file*; *c*: *Char*;

Renames file on disk

43

procedure Rename*vart*: *Text*; *consts*;

Rename file on disk

44

procedure Rename*vart*: *Text*; *p*: *PChar*;

Renames file on disk

45

procedure Rename*vart*: *Text*; *c*: *Char*;

Renames file on disk

46

procedure Reset*varf*: *file*; *l*: *LongInt*;

Opens file for reading

47

procedure Reset*varf*: *file*;

Opens file for reading

48

procedure Reset*varf*: *TypedFile*;

Opens file for reading

49

procedure Reset*vart*: *Text*;

Opens file for reading

50

procedure Rewrite*varf*: *file*; *l*: *LongInt*;

Opens file for writing

51

procedure Rewrite*varf*: *file*;

Opens file for writing

52

procedure Rewrite*varf*: *TypedFile*;

Opens file for writing

53

procedure Rewrite*vart*: *Text*;

Opens file for writing

54

procedure Seek*varf*: *file*; *Pos*: *Int*64;

Sets file position

55

function SeekEOF*vart*: *Text***:Boolean;**

Sets file position to end of file

56

function SeekEOF: Boolean;

Sets file position to end of file

57

function SeekEOLn*vart*: *Text***:Boolean;**

Sets file position to end of line

58

function SeekEOLn: Boolean;

Sets file position to end of line

59

procedure SetTextBuf*varf*: *Text*; *varBuf*;

Sets size of file buffer

60

procedure SetTextBuf*varf*: *Text*; *varBuf*; *Size*: *SizeInt*;

Sets size of file buffer

61

procedure Truncate$varF : file$**;**

Truncate the file at position

62

procedure Write$Args : Arguments$**;**

Writes variable to file

63

procedure Write$varF : Text; Args : Arguments$**;**

Write variable to file

64

procedure Writeln$Args : Arguments$**;**

Writes variable to file and append newline

65

procedure WriteLn$varF : Text; Args : Arguments$**;**

Writes variable to file and append newline