The library package Text_IO has the following declaration:

```
2       with Ada.IO_Exceptions;
package Ada.Text_IO is
3               type File_Type is limited private;
4               type File_Mode is (In_File, Out_File, Append_File);
5               type Count is range 0 .. implementation-defined;
        subtype Positive_Count is Count range 1 .. Count'Last;
        Unbounded : constant Count := 0; -- line and page length
6               subtype Field       is Integer range 0 .. implementation-defined;
        subtype Number_Base is Integer range 2 .. 16;

7               type Type_Set is (Lower_Case, Upper_Case);
8               -- File Management
9               procedure Create (File : in out File_Type;
                    Mode : in File_Mode := Out_File;
                    Name : in String    := "";
                    Form : in String    := "");
10              procedure Open   (File : in out File_Type;
                    Mode : in File_Mode;
                    Name : in String;
                    Form : in String := "");

11              procedure Close  (File : in out File_Type);
        procedure Delete (File : in out File_Type);
        procedure Reset  (File : in out File_Type; Mode : in File_Mode);
        procedure Reset  (File : in out File_Type);
12              function  Mode   (File : in File_Type) return File_Mode;
        function  Name   (File : in File_Type) return String;
        function  Form   (File : in File_Type) return String;

13              function  Is_Open(File : in File_Type) return Boolean;
14              -- Control of default input and output files
15              procedure Set_Input (File : in File_Type);
        procedure Set_Output(File : in File_Type);
        procedure Set_Error (File : in File_Type);
16              function Standard_Input  return File_Type;
        function Standard_Output return File_Type;
        function Standard_Error  return File_Type;

17              function Current_Input   return File_Type;
        function Current_Output  return File_Type;
        function Current_Error   return File_Type;
18              type File_Access is access constant File_Type;
19              function Standard_Input  return File_Access;
        function Standard_Output return File_Access;
        function Standard_Error  return File_Access;

……..
```

```
28              procedure New_Line   (File    : in File_Type;
                        Spacing : in Positive_Count := 1);
        procedure New_Line   (Spacing : in Positive_Count := 1);
29              procedure Skip_Line  (File    : in File_Type;
                        Spacing : in Positive_Count := 1);
        procedure Skip_Line  (Spacing : in Positive_Count := 1);
30              function  End_Of_Line(File : in File_Type) return Boolean;
        function  End_Of_Line return Boolean;

31              procedure New_Page   (File : in File_Type);
        procedure New_Page;
32              procedure Skip_Page  (File : in File_Type);
        procedure Skip_Page;
33              function  End_Of_Page(File : in File_Type) return Boolean;
        function  End_Of_Page return Boolean;
34              function  End_Of_File(File : in File_Type) return Boolean;
        function  End_Of_File return Boolean;
….
40              -- Character Input-Output
41              procedure Get(File : in  File_Type; Item : out Character);
        procedure Get(Item : out Character);
42              procedure Put(File : in  File_Type; Item : in Character);
        procedure Put(Item : in  Character);

43              procedure Look_Ahead (File        : in  File_Type;
                        Item        : out Character;
                        End_Of_Line : out Boolean);
        procedure Look_Ahead (Item        : out Character;
                        End_Of_Line : out Boolean);
44              procedure Get_Immediate(File      : in  File_Type;
                          Item      : out Character);
        procedure Get_Immediate(Item      : out Character);

45              procedure Get_Immediate(File      : in  File_Type;
                          Item      : out Character;
                          Available : out Boolean);
        procedure Get_Immediate(Item      : out Character;
                          Available : out Boolean);
46              -- String Input-Output
47              procedure Get(File : in  File_Type; Item : out String);
        procedure Get(Item : out String);

48              procedure Put(File : in  File_Type; Item : in String);
        procedure Put(Item : in  String);
49              procedure Get_Line(File : in  File_Type;
                     Item : out String;
                     Last : out Natural);
        procedure Get_Line(Item : out String; Last : out Natural);
50              procedure Put_Line(File : in  File_Type; Item : in String);
        procedure Put_Line(Item : in  String);
```

```
51      -- Generic packages for Input-Output of Integer Types
52              generic
                type Num is range <>;
        package Integer_IO is
53                      Default_Width : Field := Num'Width;
                Default_Base  : Number_Base := 10;
54                      procedure Get(File  : in  File_Type;
                          Item  : out Num;
                          Width : in Field := 0);
                procedure Get(Item  : out Num;
                          Width : in  Field := 0);

55                      procedure Put(File  : in File_Type;
                          Item  : in Num;
                          Width : in Field := Default_Width;
                          Base  : in Number_Base := Default_Base);
                procedure Put(Item  : in Num;
                          Width : in Field := Default_Width;
                          Base  : in Number_Base := Default_Base);
                procedure Get(From : in  String;
                          Item : out Num;
                          Last : out Positive);
                procedure Put(To   : out String;
                          Item : in Num;
                          Base : in Number_Base := Default_Base);

56              end Integer_IO;
…..

62              -- Generic packages for Input-Output of Real Types
63              generic
                type Num is digits <>;
        package Float_IO is
64                      Default_Fore : Field := 2;
                Default_Aft  : Field := Num'Digits–1;
                Default_Exp  : Field := 3;
65                      procedure Get(File  : in  File_Type;
                          Item  : out Num;
                          Width : in  Field := 0);
                procedure Get(Item  : out Num;
                          Width : in  Field := 0);

66                      procedure Put(File : in File_Type;
                          Item : in Num;
                          Fore : in Field := Default_Fore;
                          Aft  : in Field := Default_Aft;
                          Exp  : in Field := Default_Exp);
                procedure Put(Item : in Num;
                          Fore : in Field := Default_Fore;
```

```
                    Aft  : in Field := Default_Aft;
                    Exp  : in Field := Default_Exp);

67                  procedure Get(From : in String;
                    Item : out Num;
                    Last : out Positive);
            procedure Put(To   : out String;
                    Item : in Num;
                    Aft  : in Field := Default_Aft;
                    Exp  : in Field := Default_Exp);
    end Float_IO;
….

73          generic
            type Num is delta <> digits <>;
    package Decimal_IO is

74                  Default_Fore : Field := Num'Fore;
            Default_Aft  : Field := Num'Aft;
            Default_Exp  : Field := 0;
75                  procedure Get(File  : in  File_Type;
                    Item  : out Num;
                    Width : in  Field := 0);
            procedure Get(Item  : out Num;
                    Width : in  Field := 0);
76                  procedure Put(File : in File_Type;
                    Item : in Num;
                    Fore : in Field := Default_Fore;
                    Aft  : in Field := Default_Aft;
                    Exp  : in Field := Default_Exp);
            procedure Put(Item : in Num;
                    Fore : in Field := Default_Fore;
                    Aft  : in Field := Default_Aft;
                    Exp  : in Field := Default_Exp);

77                  procedure Get(From : in  String;
                    Item : out Num;
                    Last : out Positive);
            procedure Put(To   : out String;
                    Item : in Num;
                    Aft  : in Field := Default_Aft;
                    Exp  : in Field := Default_Exp);
    end Decimal_IO;
78          -- Generic package for Input-Output of Enumeration Types
79          generic
            type Enum is (<>);
    package Enumeration_IO is

80                  Default_Width   : Field := 0;
            Default_Setting : Type_Set := Upper_Case;
```

```
81                      procedure Get(File : in  File_Type;
                          Item : out Enum);
                procedure Get(Item : out Enum);
82                      procedure Put(File  : in File_Type;
                          Item  : in Enum;
                          Width : in Field    := Default_Width;
                          Set   : in Type_Set := Default_Setting);
                procedure Put(Item  : in Enum;
                          Width : in Field    := Default_Width;
                          Set   : in Type_Set := Default_Setting);

83                      procedure Get(From : in  String;
                          Item : out Enum;
                          Last : out Positive);
                procedure Put(To   : out String;
                          Item : in  Enum;
                          Set  : in  Type_Set := Default_Setting);
        end Enumeration_IO;
84      -- Exceptions
85              Status_Error : exception renames IO_Exceptions.Status_Error;
        Mode_Error   : exception renames IO_Exceptions.Mode_Error;
        Name_Error   : exception renames IO_Exceptions.Name_Error;
        Use_Error    : exception renames IO_Exceptions.Use_Error;
        Device_Error : exception renames IO_Exceptions.Device_Error;
        End_Error    : exception renames IO_Exceptions.End_Error;
        Data_Error   : exception renames IO_Exceptions.Data_Error;
        Layout_Error : exception renames IO_Exceptions.Layout_Error;
private
        ... -- not specified by the language
end Ada.Text_IO;
```