FUTURE VISION BIE

One Stop for All Study Materials

& Lab Programs

Future Vision

By K B Hemanth Raj

Scan the QR Code to Visit the Web Page

Or

Visit : https://hemanthrajhemu.github.io

Gain Access to All Study Materials according to VTU,
CSE – Computer Science Engineering,
ISE – Information Science Engineering,
ECE - Electronics and Communication Engineering
& MORE...

Join Telegram to get Instant Updates: https://bit.ly/VTU_TELEGRAM

Contact: MAIL: futurevisionbie@gmail.com

INSTAGRAM: www.instagram.com/hemanthraj_hemu/

INSTAGRAM: www.instagram.com/futurevisionbie/

WHATSAPP SHARE: https://bit.ly/FVBIESHARE

1. **Write a C++ program to read series of names, one per line, from standard input and write these names spelled in reverse order to the standard output using I/O redirection and pipes. Repeat the exercise using an input file specified by the user instead of the standard input and using an output file specified by the user instead of the standard output.**

I/O redirection

Operating systems provide shortcuts for switching between standard I/O(stdin and stdout) and regular file I/O. I/O redirection is used to change a program so it writes its output to a regular file rather than to stdout.

- In both DOS and UNIX, the standard output of a program can be redirected to a file with the > symbol.
- In both DOS and UNIX, the standard input of a program can be redirected to a file with the < symbol.
- The notations for input and output redirection on the command line in Unix are

```
< file                (redirect stdin to "file")
> file                (redirect stdout to "file")
```

- Example:

```
list.exe > myfile
```

The output of the executable file is redirected to a file called “myfile”

pipe

- Piping: using the output of one program as input to another program. A connection between standard output of one process and standard input of a second process.
- In both DOS and UNIX, the standard output of one program can be piped (connected) to the standard input of another program with the | symbol.
- Example:

```
program1 | program2
```

- Output of program1 is used as input for program2

File I/O:

perform output and input of characters to or from files

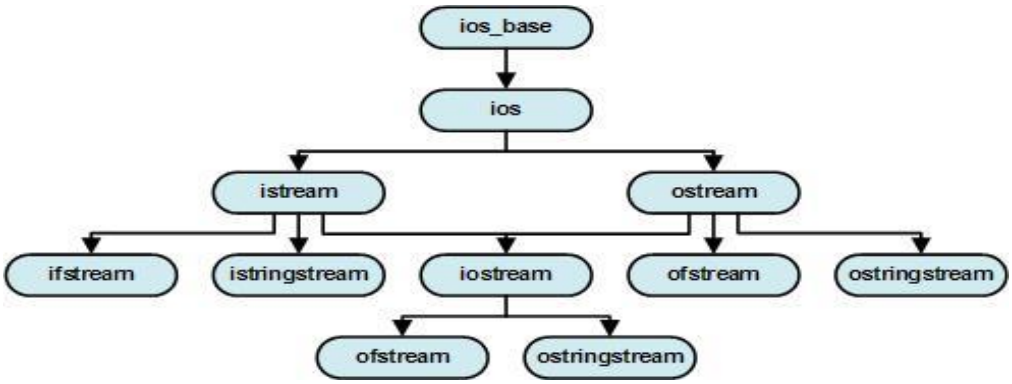
Standerd I/O:

- standard streams are preconnected input and output channels between a computer program and its environment (typically a text terminal) when it begins execution. The three I/O connections are called standard input (stdin), standard output (stdout) and standard error (stderr).

fstream

- fstream provides an interface to read and write data from files as input/output streams.The file to be associated with the stream can be specified either as a parameter in the constructor or by calling member open.

- After all necessary operations on a file have been performed, it can be closed (or disassociated) by calling member close. Once closed, the same file stream object may be used to open another file.



Function to open a file:

The first operation generally performed on an object of one of these classes is to associate it to a real file. This procedure is known as to *open a file*. An open file is represented within a program by a stream object (an instantiation of one of these classes, in the previous example this was `myfile`) and any input or output operation performed on this stream object will be applied to the physical file associated to it.

In order to open a file with a stream object we use its member function `open()`:

```
open (filename, mode);
```

Where `filename` is a null-terminated character sequence of type `const char *` (the same type that string

literals have) representing the name of the file to be opened, and `mode` is an optional parameter with a combination of the following flags:

ios::in	Open for input operations.
ios::out	Open for output operations.
ios::binary	Open in binary mode.
ios::ate	Set the initial position at the end of the file. If this flag is not set to any value, the initial position is the beginning of the file.
ios::app	All output operations are performed at the end of the file, appending the content to the current content of the file. This flag can only be used in streams open for output-only operations.
ios::trunc	If the file opened for output operations already existed before, its previous content is deleted and replaced by the new one.

Function to Closing the Files

To disassociate a logical program file from a physical system file.

- **Prototypes:**

  int close (int Handle);

- **Example:**

  close (Input);

Getline Function:

Extracts characters from specified location and stores them into *str* until the delimitation character *delim* is found or length equal to size.

- **prototype**

  fstream str;
  Str.getline (istream& is, int size, char delim);

Program 1

```
#include<iostream.h>
#include<stdio.h>
#include<string.h>
#include<fstream.h>
#include<conio.h>
#include<iomanip.h>
#include<stdlib.h>

class std_file
{
private:
      char name[10][20];
      char
input[20],output[20],str[20]; public:
      void std_io();
      void file_io();
};

void std_file::std_io(){
int n,i;
cout<<"enter the number of names to read
           "<<endl; cin>>n;
           cout<<"enter the names"<<endl;
           for(i=0;i<n;i++)
           gets(name[i]);
           cout<<"the reversed names are"<<endl;
           for(i=0;i<n;i++)
             cout<<strrev(name[i])<<endl;

}
 void std_file::file_io(){
      fstream ifile,ofile;
      cout<<"enter the filename which contain list of names"<<endl;
      cin>>input;
      ifile.open(input,ios::in);
      if(!ifile)
      {
            cout<<"file doesnot exist";
            exit(0);
            getch();
      }
      cout<<"enter the filename to store names in reverse order"<<endl;
      cin>>output;
      ofile.open(output,ios::out);
      while(!ifile.eof())
      {
            ifile.getline(str,20);
            ofile<<strrev(str)<<endl; //to reverse string characters
      }
}
```

```
void main()
{
      int i,num,len;
      std_file s;
      clrscr();
      for(;;)
      {
      cout<<"1:file i/o\n2:standard i/o\noany other to exit\n";
      cin>>num;
      switch(num){
      case 1:
            s.std_io();
            break;

      case 2:
            s.file_io();
            break;
      default:
            exit(0);
                  }
     }
}
```

Output 1:

1: file i/o
2: standard i/o
0 : any other to exit
1
enter the number of names to read
2
Enter the names
keerthana
madhu
The reversed names are
anahtreek
uhdam

1: file i/o
2: standard i/o
0: any other to exit
2
Enter the filename which contain list of names
student.txt
Enter the filename to store names in reverse
order studentr.txt

1:file i/o
2:standard i/o

any other to exit
0
c:\tc>type student.txt
ragu
navya

c:\tc>type studentr.dat
ugar
ayvan

Output 2:

1:file i/o
2:standard i/o
any other to exit
2
enter the filename which contain list of names
std.txt
file does not exist

1:file i/o
2:standard i/o
any other to exit
0

Output 3: using I/O redirection and pipes (Run the program in Command prompt)

I/O redirection : Redirect the output from *stdout* to a file aaa.txt
Syntax : program1 >filename
Exp:
c:\tc>fslab1 > aaa.txt
1
3
keerthana
madhu
bavana
1:file i/o
2:standard i/o
any other to exit
0
c:\tc>type aaa.txt
1:file i/o
2:standard i/o

any other to exit
enter the number of names to read
enter the names
the reversed names are
anahtreek
uhdam
anavab
1:file i/o
2:standard i/o
any other to exit

Pipes :take any *stdout* output from program 1 and use it in place of any *stdin* input to program2. Syntax : program1 | program 2

c:\tc>type student.txt | sort
bavana
keerthana
madhu