**C++**

**Project File**

Name: **Naveen Venkat**

Roll: **9169946**

Class: **12-F**

Session: **2013-2014**

School: **Mount Carmel, anand niketan**

**DevR**

**DevELOPER REDEFINED**

WORD PROCESSOR

NAME: **Naveen Venkat**

CLASS: **12 - F**

INSTITUTION: **Mount Carmel School**

This is certified to be the bonafide work of ***Naveen Venkat*** in the subject ***Computer Science*** during the academic year ***2013-2014***.

-------------------------

Teacher In-charge

(Mr. Sanjeev Sharma)

---------------------------

Examiner’s Signature

-----------------

Date

/\*==============================================================================

DEVR version 1.6

CREATED AND DEVELOPED BY

NAVEEN VENKAT

================================================================================

***DevR is a word processor which can be used to work with text files. Files can  
be viewed, created, deleted, appended, encrypted and decrypted.  
A history of all the commands is saved in a file called 'core'. It can be  
prevented by using command '\_INCOGNITO\_1'. Once on, no history of the commands  
typed will be saved. This can be used for anonymity.  
Some other features include the ability to manage a list of important files in  
a library One can also change the color of the console window to select the best  
suiting scheme.***

================================================================================

TWITTER: twitter.com/ServerProcessor

EMAIL: nmakes@outlook.com

================================================================================

**RECOMMENDED MINIMUM SYSTEM REQUIREMENTS**

OS: Windows Platform

CPU: 1.0 GHz (16bit, 32bit and 64bit supported)

RAM: At least 1MB (requires more when handling files)

VIDEO: Standard VGA Monitor

STORAGE: 200 KB

PERIPHERAL: Keyboard

==============================================================================\*/

**// HEADER FILES**

#include <fstream.h>

#include <stdio.h>

#include <conio.h>

#include <string.h>

#include <process.h>

#include <stdlib.h>

#include <ctype.h>

**// GLOBAL VARIABLES**

float version=1.6;

fstream REGISTRY;

fstream HISTORY("core",fstream::app);

char INPUT[128];

char USER[32];

char EXE[11]="devr16.exe";

char APP\_NAME[4]="devr";

int CONSOLE\_WIDTH = 80;

char CONSOLE\_COLOR[9] = "Color 1F";

int INIT=0;

int INCOGNITO=0;

int HIST=1;

int CRYPT=0;

**// FUNCTION DECLARATIONS**

void CALL\_CLR();

void CALL\_HELP();

void CALL\_CREATE();

void CALL\_CLOSE();

void CALL\_OPEN();

void CALL\_DELETE();

void CALL\_INCOGNITO(int);

void CALL\_COPY();

void CALL\_MEMORY();

int CALL\_UNINSTALL\_DEVR();

void CALL\_SWITCH\_USER();

void CALL\_ENCRYPT();

void CALL\_DECRYPT();

void CALL\_CONSOLE();

void CALL\_COLOR();

void CALL\_REFRESH();

void CALL\_LIBRARY();

void CALL\_ABOUT();

void login();

void core(char\*);

int corecheck(char\*);

**// LIBRARY STRUCTURE (used in library)**

struct \_library

{

char title[20];

char path[64];

};

**// CHECK() FUNCTION (understands and translates user interrupts)**

void check()

{

if(HIST)

{

core(INPUT);

}

if(!strcmp(INPUT,"\_CLR"))

{ CALL\_CLR(); }

else if(!strcmp(INPUT,"\_INCOGNITO\_1"))

{ CALL\_INCOGNITO(1); }

else if(!strcmp(INPUT,"\_INCOGNITO\_0"))

{ CALL\_INCOGNITO(0); }

else if(!strcmp(INPUT,"\_CREATE"))

{ CALL\_CREATE(); }

else if(!strcmp(INPUT,"\_OPEN"))

{ CALL\_OPEN(); }

else if(!strcmp(INPUT,"\_CLOSE"))

{ CALL\_CLOSE(); }

else if(!strcmp(INPUT,"\_COPY"))

{ CALL\_COPY(); }

else if(!strcmp(INPUT,"\_HELP"))

{ CALL\_HELP(); }

else if(!strcmp(INPUT,"\_DELETE"))

{ CALL\_DELETE(); }

else if(!strcmp(INPUT,"\_MEMORY"))

{ CALL\_MEMORY(); }

else if(!strcmp(INPUT,"\_UNINSTALL\_DEVR"))

{ CALL\_UNINSTALL\_DEVR(); }

else if(!strcmp(INPUT,"\_SWITCH\_USER"))

{ CALL\_SWITCH\_USER(); }

else if(!strcmp(INPUT,"\_ENCRYPT"))

{ CALL\_ENCRYPT(); }

else if(!strcmp(INPUT,"\_DECRYPT"))

{ CALL\_DECRYPT(); }

else if(!strcmp(INPUT,"\_CONSOLE"))

{ CALL\_CONSOLE(); }

else if(!strcmp(INPUT,"\_COLOR"))

{ CALL\_COLOR(); }

else if(!strcmp(INPUT,"\_REFRESH"))

{ CALL\_REFRESH(); }

else if(!strcmp(INPUT,"\_LIBRARY"))

{ CALL\_LIBRARY(); }

else if(!strcmp(INPUT,"\_ABOUT"))

{ CALL\_ABOUT(); }

else

{

if(INIT==1)

{

REGISTRY << INPUT << endl;

}

}

}

**// MAIN() FUNCTION**

**int main()**

**{**

**login();**

**CALL\_CLR();**

**HISTORY << endl << "\t[STARTED BY: " << USER << "]\n";**

**cout << "DevR - Development Redefined | version " << version;**

**cout << endl;**

**cout << "Type \_HELP for help.";**

**cout << endl;**

**for(int i=1; i<=CONSOLE\_WIDTH; i++)**

**{**

**cout << '-';**

**}**

**cout << endl;**

**do**

**{**

**cout << INIT << " >> ";**

**gets(INPUT);**

**check();**

**} while(strcmp(INPUT,"\_EXIT"));**

**HISTORY.close();**

**if(INIT==1)**

**{**

**REGISTRY.close();**

**}**

**CALL\_REFRESH();**

**system("color 07");**

**return(0);**

**}**

**// FUNCTION DEFINITIONS**

**// [1] CALL\_CREATE (pertaining to ‘\_CREATE’ command)**

void CALL\_CREATE()

{

if(INIT==1)

{

cout << "\*ERROR\* Please close the current file before initialising a new file!\a";

cout << endl;

}

else

{

cout << "- Enter the path of the file: ";

gets(INPUT);

core(INPUT);

if(!strcmp(INPUT,""))

{

cout << "\*ERROR\* Please enter a valid file path!\a";

cout << endl;

}

else if(!strcmp(INPUT,"\_CANCEL"))

{

cout << "- File Creation Cancelled!\a" << endl;

}

else if(corecheck(INPUT) && (strcmp(USER,"\_DEBUG")))

{

cout << "\*ERROR\* Cannot override the core file!\a";

cout << endl;

}

else if(!strcmp(INPUT,EXE) && (strcmp(USER,"\_DEBUG")))

{

cout << "\*ERROR\* Cannot override devr!\a";

cout << endl;

}

else

{

REGISTRY.open(INPUT,fstream::app);

INIT = 1;

if(REGISTRY.bad())

{

cout << "\*ERROR\* Path not found!\a" << endl;

REGISTRY.close();

INIT = 0;

}

else

{

cout << "- INITIALISATION SUCCESSFUL!";

cout << endl << "- WORKING ON FILE: " << INPUT << endl;

cout << endl << "ALL THE KEYWORDS YOU TYPE WILL BE WRITTEN INTO THE FILE. TO STOP TYPE \_CLOSE" << endl;

}

}

}

}

**// [2] CALL\_CLOSE (pertaining to ‘\_CLOSE’ command)**

void CALL\_CLOSE()

{

if(INIT==1){REGISTRY.close();}

INIT = 0;

}

**// [3] CALL\_HELP (pertaining to ‘\_HELP’ command)**

void CALL\_HELP()

{

CALL\_CLR();

cout << endl;

cout << "\t THE FOLLOWING ARE THE STANDARD COMMANDS IN THE DevR CONSOLE" << endl;

cout << endl;

cout << "\_CREATE: Create a file (write data into it)." << endl;

cout << "\_CLOSE: Close an open file." << endl;

cout << "\_OPEN: Open a file to read data from it." << endl;

cout << "\_COPY: Copy a file into another file." << endl;

cout << "\_DELETE: Delete a file." << endl;

cout << "\_CANCEL: Cancel an ongoing operation." << endl;

cout << endl;

cout << "\_ENCRYPT: Encrypt a file." << endl;

cout << "\_DECRYPT: Decrypt an encrypted file." << endl;

cout << endl;

cout << "\_LIBRARY: View the library and related functions." << endl;

cout << "\_CONSOLE: View Console related commands (Color, About, Memory)." << endl;

cout << endl;

cout << "\_CLR: Clear the screen." << endl;

cout << "\_REFRESH: Refresh the console." << endl;

cout << endl;

cout << "\_INCOGNITO\_1: Enable INCOGNITO mode (nothing will be saved in history)." << endl;

cout << "\_INCOGNITO\_0: Disable INCOGNITO mode." << endl;

cout << endl;

cout << "\_SWITCH\_USER: Change the current user (username ‘\_DEBUG’ gives complete access)." << endl;

cout << "\_HELP: Display this help screen." << endl;

cout << "\_EXIT: Exit devr." << endl << endl;

cout << endl;

cout << "DevR version " << version << "\nCreated and Developed by Naveen Venkat\nTwitter: twitter.com/ServerProcessor\nEmail: nmakes@outlook.com" << endl << endl;

cout << "\t\tNOTE: Please scroll up to view more commmands." << endl << endl;

}

**// [4] CALL\_OPEN (pertaining to ‘\_OPEN’ command)**

void CALL\_OPEN()

{

if(INIT==1)

{

cout << "\*ERROR\* Please close the current file before opening a new file!\a";

cout << endl;

}

else if(INIT==0)

{

cout << "- Enter the path of the file: ";

gets(INPUT);

core(INPUT);

if(!strcmp(INPUT,""))

{

cout << "\*ERROR\* Please enter a valid file path!\a";

cout << endl;

}

else if(!strcmp(INPUT,"\_CANCEL"))

{

cout << "- File Opening Cancelled!\a" << endl;

}

else if(corecheck(INPUT)&&(strcmp(USER,"\_DEBUG")))

{

cout << "\*ERROR\* Protected Command!\a";

cout << endl;

}

else

{

REGISTRY.open(INPUT,fstream::in);

INIT = 1;

if(!REGISTRY.good())

{

cout << "\*ERROR\* The file does not exist!\a" << endl;

}

else

{

CALL\_CLR();

char ch;

while(REGISTRY.eof()!=1)

{

REGISTRY.get(ch);

cout << ch;

}

cout << endl;

for(int i=1; i<=CONSOLE\_WIDTH; i++)

{

cout << '-';

}

cout << endl;

}

REGISTRY.close();

INIT = 0;

}

}

}

**// [5] CALL\_DELETE (pertaining to ‘\_DELETE’ command)**

void CALL\_DELETE()

{

if(INIT==1)

{

cout << "\*ERROR\* Please close the current file before deleting a file!\a" << endl;

}

else if(!strcmp(USER,""))

{

cout << "\*ERROR\* Protected Command!\a" << endl;

}

else

{

cout << "- Enter the path of the file: ";

gets(INPUT);

core(INPUT);

if(!strcmp(INPUT,""))

{

cout << "\*ERROR\* Please enter a valid file path!\a";

cout << endl;

}

else if(!strcmp(INPUT,"\_CANCEL"))

{

cout << "- File Deletion Cancelled!\a" << endl;

}

else if( corecheck(INPUT) && strcmp(USER,"\_DEBUG") )

{

cout << "\*ERROR\* Cannot delete the core file!\a";

cout << endl;

}

else if(!strcmp(INPUT,EXE) && (strcmp(USER,"\_DEBUG")))

{

cout << "\*ERROR\* Please use \_UNINSTALL\_DEVR command to uninstall devr!\a";

cout << endl;

}

else

{

REGISTRY.open(INPUT,fstream::in);

INIT=1;

if(!REGISTRY.good())

{

cout << "\*ERROR\* The file does not exist!\a" << endl;

REGISTRY.close(); INIT = 0;

}

else

{

REGISTRY.close(); INIT = 0;

if (corecheck(INPUT) && !strcmp(USER,"\_DEBUG"))

{

HISTORY.close(); HIST = 0;

}

remove(INPUT);

cout << "- Successfully Deleted!";

cout << endl;

}

}

}

}

**// [6] LOGIN (the first function to be called**

void login()

{

CALL\_CLR();

cout << "\n devr " << version;

cout << "\n Created and Developed by Naveen Venkat";

cout << "\n nmakes@outlook.com";

cout << "\n\n\n\n\n\n\n\n\n\t\t\tLOGIN USERNAME: ";

gets(USER);

}

**// [7] CALL\_INCOGNITO (pertaining to ‘\_INCOGNITO\_0’ and ‘\_INCOGNITO\_1’ commands)**

void CALL\_INCOGNITO(int x)

{

if(x)

{

if(INCOGNITO==1)

{

cout << "\*ERROR\* INCOGNITO MODE is already activated!\a" << endl;

}

else

{

INCOGNITO=1;

cout << "\n- INCOGNITO MODE ACTIVATED. \n- Text and commands that you type WILL NOT BE saved anywhere in the history.";

cout << "\n- However, it will not affect the files that you edit.";

cout << "\n- Be wary of:";

cout << "\n\tMalicious software that tracks your keystrokes";

cout << "\n\tSurveillance by secret agents";

cout << "\n\tPeople standing behind you\n\n";

}

}

else

{

INCOGNITO=0;

cout << "\n- INCOGNITO MODE OFF." << endl;

cout << "- Text and commands that you type WILL BE saved in the history.\n\n";

}

}

**// [8] CALL\_COPY (pertaining to ‘\_COPY’ command)**

void CALL\_COPY()

{

fstream TEMPFILE; char RAM[128];

if(INIT==1)

{

cout << "\*ERROR\* Please close the current file before copying a file!\a";

cout << endl;

}

else if(!strcmp(USER,""))

{

cout << "\*ERROR\* Protected Command!\a" << endl;

}

else if(INIT==0)

{

cout << "- Enter the path of the source file: ";

gets(INPUT);

core(INPUT);

if(!strcmp(INPUT,""))

{

cout << "\*ERROR\* Please enter a valid file path!\a" << endl;

}

else if(!strcmp(INPUT,"\_CANCEL"))

{

cout << "- File Copying Cancelled!\a" << endl;

}

else

{

REGISTRY.open(INPUT,fstream::in);

if(!REGISTRY.good())

{

cout << "\*ERROR\* Source file not found!\a" << endl;

REGISTRY.close(); INIT = 0;

}

else

{

REGISTRY.close(); INIT = 0;

cout << "- Enter the path of the destination file: ";

gets(RAM);

core(RAM);

if(!strcmp(RAM,""))

{

cout << "\*ERROR\* Please enter a valid file path!\a";

cout << endl;

}

else if(!strcmp(RAM,"\_CANCEL"))

{

cout << "- File Copying Cancelled!\a" << endl;

REGISTRY.close(); INIT=0;

}

else

{

REGISTRY.open(INPUT,fstream::in); INIT = 1;

TEMPFILE.open(RAM,fstream::out);

if(TEMPFILE.bad())

{

cout << "\*ERROR\* Destination path not found!\a" << endl;

}

else

{

while(REGISTRY.eof()!=1)

{

REGISTRY.getline(INPUT,128);

TEMPFILE << INPUT << endl;

}

cout << "- Successfully Copied!" << endl;

}

REGISTRY.close(); INIT = 0;

TEMPFILE.close();

}

}

}

}

}

**// [9] CALL\_MEMORY (pertaining to ‘\_MEMORY’ command)**

void CALL\_MEMORY()

{

cout << "\n\*RAM INFO\*" << endl;

cout << "- [" << sizeof(version) << "B] version: " << version;

cout << "\n- [" << sizeof(INPUT) << "B] INPUT: " << INPUT;

cout << "\n- [" << sizeof(USER) << "B] USER: " << USER;

cout << "\n- [" << sizeof(INIT) << "B] INIT: " << INIT;

cout << "\n- [" << sizeof(INCOGNITO) << "B] INCOGNITO: " << INCOGNITO;

cout << "\n- [" << sizeof(CONSOLE\_COLOR) << "B] CONSOLE\_COLOR: " << CONSOLE\_COLOR;

cout << endl << endl << "\a";

}

int CALL\_UNINSTALL\_DEVR()

{

char ch;

cout << "\n- ARE YOU SURE TO UNINSTALL DEVR? (Y/N): ";

cin >> ch;

if(ch=='y'||ch=='Y')

{

cout << "- Initiating Self Destruct Mode";

REGISTRY.close();

HISTORY.close();

cout << "\n- Deleting files: " << endl;

cout << "\t+ core";

remove("core");

cout << "\t+ library";

remove("library.set");

cout << "\n\n\t\t\tR.I.P devr\n\tA peaceful goodbye. What else can we say.\n\t Thank you for using devr, by the way!";

cout << endl << endl;

cout << "You can continue working. When you want to stop type \_EXIT";

cout << endl;

}

return(0);

}

**// [10] core (this controls the flow of data into history file)**

void core(char a[])

{

if(!INCOGNITO)

{

HISTORY << "[" << INIT << "] " << a << endl;;

}

else

{

HISTORY << "[" << INIT << "] " << "<{[INCOGNITO]}>" << endl;

}

}

**// [11] corecheck (scans the input for any valid combination of ‘core’)**

int corecheck(char a[])

{

if(strlen(a)==4)

{

if( (a[0]=='c' || a[0]=='C') && (a[1]=='o' || a[1]=='O') &&

(a[2]=='r' || a[2]=='R') && (a[3]=='e' || a[3]=='E') )

{

return 1;

}

else

{ return 0; }

}

else

{ return 0; }

}

**// [11] CALL\_SWITCH\_USER (pertaining to ‘\_SWITCH\_USER’ command)**

void CALL\_SWITCH\_USER()

{

char \*RAM = new char[32];

cout << "- Enter the new user name: ";

strcpy(RAM,USER);

gets(USER);

core(USER);

cout << "- User successfully changed from '" << RAM << "' to '" << USER << "'" << endl;

}

**// [12] CALL\_ENCRYPT (pertaining to ‘\_ENCRYPT’ command)**

void CALL\_ENCRYPT()

{

if(INIT==1)

{

cout << "\*ERROR\* Please close the current file before encrypting a file!\a";

cout << endl;

}

else if(INIT==0)

{

cout << "- Enter the path of the file to encrypt: ";

gets(INPUT);

core(INPUT);

if(!strcmp(INPUT,""))

{

cout << "\*ERROR\* Please enter a valid file path!\a";

cout << endl;

}

else if(!strcmp(INPUT,"\_CANCEL"))

{

cout << "- File Encryption Cancelled!\a" << endl;

}

else if(corecheck(INPUT)&&(strcmp(USER,"\_DEBUG")))

{

cout << "\*ERROR\* Protected Command!\a";

cout << endl;

}

else

{

REGISTRY.open(INPUT,fstream::in);

INIT = 1;

if(!REGISTRY.good())

{

cout << "\*ERROR\* The file does not exist!\a" << endl;

REGISTRY.close();

INIT=0;

}

else

{

char LOC[128];

cout << "- Enter the path to save the encrypted file: ";

gets(LOC);

core(LOC);

if(!strcmp(LOC,"\_CANCEL"))

{

cout << "- File Encryption Cancelled!\a" << endl;

REGISTRY.close(); INIT=0;

}

else

{

ofstream TEMPexh(LOC, fstream::binary);

if (TEMPexh.bad())

{

cout << "\*ERROR\* The path does not exist!\a" << endl;

TEMPexh.close();

}

else

{

int l=0;

struct EXPRESS

{

int no;

char ch;

} XT;

while(REGISTRY.eof()!=1)

{

REGISTRY.get(XT.ch);

if (REGISTRY.eof()==0)

{

randomize();

XT.no = random(64);

XT.ch = XT.ch + XT.no;

TEMPexh.write((char\*)&XT, sizeof(XT));

l++;

}

}

TEMPexh.close();

cout << "- Encrypted successfully! " << l << " bytes of information translated!" << endl;

}

}

}

REGISTRY.close();

INIT = 0;

}

}

}

**// [13] CALL\_DECRYPT (pertaining to ‘\_DECRYPT’ command)**

void CALL\_DECRYPT()

{

if(INIT==1)

{

cout << "\*ERROR\* Please close the current file before decrypting a file!\a";

cout << endl;

}

else if(INIT==0)

{

cout << "- Enter the path of the file to decrypt: ";

gets(INPUT);

core(INPUT);

if(!strcmp(INPUT,""))

{

cout << "\*ERROR\* Please enter a valid file path!\a";

cout << endl;

}

else if(!strcmp(INPUT,"\_CANCEL"))

{

cout << "- File Decryption Cancelled!\a" << endl;

}

else if(corecheck(INPUT)&&(strcmp(USER,"\_DEBUG")))

{

cout << "\*ERROR\* Protected Command!\a";

cout << endl;

}

else

{

REGISTRY.open(INPUT,fstream::in|fstream::binary);

INIT = 1;

if(!REGISTRY.good())

{

cout << "\*ERROR\* The file does not exist!\a" << endl;

REGISTRY.close(); INIT=0;

}

else

{

char LOC[128];

cout << "- Enter the path to save the decrypted file: ";

gets(LOC);

core(LOC);

if(!strcmp(LOC,"\_CANCEL"))

{

cout << "- File Decryption Cancelled!\a" << endl;

REGISTRY.close(); INIT=0;

}

else

{

ofstream TEMPexh(LOC);

if (TEMPexh.bad())

{

cout << "\*ERROR\* The path does not exist!\a" << endl;

TEMPexh.close();

}

else

{

int l=0;

struct EXPRESS

{

int no;

char ch;

} XT;

randomize();

while(REGISTRY.eof()!=1)

{

REGISTRY.read((char\*)&XT, sizeof(XT));

if(REGISTRY.eof()==0)

{

XT.ch = XT.ch - XT.no;

TEMPexh.put(XT.ch);

l++;

}

}

TEMPexh.close();

cout << "- Decrypted successfully! " << l << " bytes of information translated!" << endl;

}

TEMPexh.close();

}

}

REGISTRY.close(); INIT = 0;

}

}

}

**// [14] CALL\_CLR (pertaining to ‘\_CLR’ command)**

void CALL\_CLR()

{

clrscr();

system(CONSOLE\_COLOR);

}

**// [15] CALL\_CONSOLE (pertaining to ‘\_CONSOLE’ command)**

void CALL\_CONSOLE()

{

cout << "\n";

cout << "- SELECT A COLOR SCHEME FOR THE CONSOLE (\_COLOR)." << endl;

cout << "- SHOW THE CURRENT RAM USAGE (\_MEMORY)." << endl;

cout << "- KNOW ABOUT DEVR CONSOLE (\_ABOUT)." << endl;

cout << "- UNINSTALL DEVR - say goodbye forever (\_UNINSTALL\_DEVR)." << endl << endl;

}

**// [16] CALL\_COLOR (pertaining to ‘\_COLOR’ command)**

void CALL\_COLOR()

{

cout << endl;

char TEMPest[3]; int flag=0;

cout << "- Enter a combination of two charecters." << endl;

cout << "- The first charecter: Background color." << endl;

cout << "- The second charecter: Text color." << endl;

cout << "\n\t0 = Black\t\t8 = Gray";

cout << "\n\t1 = Blue\t\t9 = Light Blue";

cout << "\n\t2 = Green\t\tA = Light Green";

cout << "\n\t3 = Aqua\t\tB = Light Aqua";

cout << "\n\t4 = Red\t\t\tC = Light Red";

cout << "\n\t5 = Purple\t\tD = Light Purple";

cout << "\n\t6 = Yellow\t\tE = Light Yellow";

cout << "\n\t7 = White\t\tF = Bright White";

cout << "\n\nFor example, the blue-white combination is 1F";

cout << endl << endl << "Enter a combination or '\_C' to cancel: ";

gets(TEMPest);

core(TEMPest);

if(!strcmp(TEMPest,"\_C"))

{

cout << "- Console Color Changing Cancelled!\a" << endl << endl;

}

else if( isdigit(TEMPest[0]) || (TEMPest[0]>='A'&&TEMPest[0]<='F') || (TEMPest[0]>='a'&&TEMPest[0]>='f') )

{

CONSOLE\_COLOR[6] = TEMPest[0];

flag=1;

if( isdigit(TEMPest[1]) || (TEMPest[1]>='A'&&TEMPest[1]<='F') || (TEMPest[0]>='a'&&TEMPest[0]>='f') )

{

CONSOLE\_COLOR[7] = TEMPest[1];

flag=1;

}

else

{

flag=0;

}

}

else

{

flag=0;

}

if(flag==1)

{

cout << "\n- Press any key to change the color... ";

getch();

CALL\_CLR();

}

else if(!strcmp(TEMPest,"\_C"))

{

// Nothing to be done here

}

else

{

cout << "\*ERROR\* Please enter a combination of 0-9 and A-F or \_C to cancel!\a" << endl << endl;

}

}

**// [17] CALL\_REFRESH (pertaining to ‘\_REFRESH’ command)**

void CALL\_REFRESH()

{

gotoxy(1,1);

for(int i=1; i<=24000; i++)

{

cout << ' ';

}

gotoxy(1,1);

CALL\_CLR();

cout << "DevR - Development Redefined | version " << version;

cout << endl;

cout << "Type \_HELP for help.";

cout << endl;

for(int j=1; j<=CONSOLE\_WIDTH; j++)

{

cout << '-';

}

cout << endl;

}

**// [18] CALL\_LIBRARY (pertaining to ‘\_LIBRARY’ command)**

void CALL\_LIBRARY()

{

int n, N, L, i;

int flag=0; //for delete function

do

{

CALL\_CLR();

cout << endl << "\*LIBRARY\*" << endl;

cout << "[1] Show collections" << endl;

cout << "[2] Add a book" << endl;

cout << "[3] Delete a book" << endl;

cout << "[4] Back to programming" << endl;

cout << endl << "- Enter an option: ";

cin >> n;

fstream TEMPcol;

fstream temploc;

\_library \_unit;

switch(n)

{

case 1:

TEMPcol.open("library.set", ios::in|ios::binary);

if(TEMPcol.bad())

{

cout << "\*ERROR\* No collection exists!\a" << endl;

TEMPcol.close();

}

else

{

i=1;

cout << endl;

cout << "Sno\tTitle (Path)" << endl;

cout << "----------------------" << endl;

while(TEMPcol)

{

TEMPcol.read((char\*)&\_unit, sizeof(\_unit));

if(!TEMPcol.eof())

{

cout << i << "\t" << \_unit.title << " (" << \_unit.path << ")" << endl;

i++;

}

}

}

TEMPcol.close();

cout << endl << "Press any key to continue...";

getch();

break;

case 2:

TEMPcol.open("library.set", ios::app|ios::binary);

if(TEMPcol.bad())

{

cout << "\*ERROR\* Could not add book. If this problem persists, please contact your vendor.\a" << endl;

TEMPcol.close();

}

else

{

cout << "- Please enter the TITLE and the PATH of the directory where the book is saved.";

cout << "\n- TITLE: ";

gets(\_unit.title);

if(!strcmp(\_unit.title, "\_CANCEL"))

{

cout << "- Book addition cancelled!\a" << endl;

}

else

{

cout << "- PATH: ";

gets(\_unit.path);

if(!strcmp(\_unit.path,"\_CANCEL"))

{

cout << "- Book addition cancelled!\a" << endl;

}

else

{

TEMPcol.write((char\*)&\_unit, sizeof(\_unit));

cout << "\n- Successfully saved!" << endl;

}

}

}

TEMPcol.close();

cout << endl << "Press any key to continue...";

getch();

break;

case 3:

flag = 0;

TEMPcol.open("library.set", ios::in|ios::binary);

if(TEMPcol.bad())

{

cout << "\*ERROR\* No collection exists!\a" << endl;

TEMPcol.close();

}

else

{

i=1;

cout << endl;

cout << "Sno\tTitle (Path)" << endl;

cout << "----------------------" << endl;

while(TEMPcol)

{

TEMPcol.read((char\*)&\_unit, sizeof(\_unit));

if(!TEMPcol.eof())

{

cout << i << "\t" << \_unit.title << " (" << \_unit.path << ")" << endl;

i++;

}

}

}

TEMPcol.close();

cout << "\n- Enter the SNo. of the book to delete: ";

cin >> N;

temploc.open("tempbuf.set", ios::out|ios::binary);

TEMPcol.open("library.set", ios::in|ios::binary);

for(L=1; (!TEMPcol.eof())&&(flag==0); L++)

{

TEMPcol.read((char\*)&\_unit, sizeof(\_unit));

if(L==N)

{

flag=1;

}

else

{

temploc.write((char\*)&\_unit, sizeof(\_unit));

}

}

for(L=1; !TEMPcol.eof(); L++)

{

TEMPcol.read((char\*)&\_unit, sizeof(\_unit));

if(!TEMPcol.eof())

{

temploc.write((char\*)&\_unit, sizeof(\_unit));

}

}

TEMPcol.close(); temploc.close();

if(flag==0)

{

remove("tempbuf.set");

cout << "\n\*ERROR\* Book not found!\a" << endl;

}

else

{

remove("library.set");

rename("tempbuf.set","library.set");

}

TEMPcol.close();

temploc.close();

cout << endl << "Press any key to continue...";

getch();

break;

case 4:

break;

default:

cout << "\n\*ERROR\* Wrong choice\a" << endl;

n=0;

cout << "\nPress any key to continue...";

getch();

}

} while(n!=4);

cout << endl;

}

**// [19] CALL\_ABOUT (pertaining to ‘\_ABOUT’ command)**

void CALL\_ABOUT()

{

CALL\_CLR();

cout << endl << endl;

cout << "================================================================================";

cout << endl;

cout << " DEVR version 1.6";

cout << endl;

cout << " CREATED AND DEVELOPED BY";

cout << endl;

cout << " NAVEEN VENKAT";

cout << endl << endl;

cout << "================================================================================";

cout << endl;

cout << "\t\t\t\t.::A SHORT SUMMARY::." << endl << endl;

cout << "DevR is a word processor which can be used to work with text files. Files can" << endl;

cout << "be viewed, created, deleted, appended, encrypted and decrypted." << endl;

cout << "A history of all the commands is saved in a file called 'core'. It can be" << endl;

cout << "prevented by using command '\_INCOGNITO\_1'. Once on, no history of the commands" << endl;

cout << "typed will be saved. This can be used for anonymity." << endl;

cout << "Some other features include the ability to manage a list of important files in" << endl;

cout << "a library. One can also change the color of the console window to select the\n”;

cout << "best suiting scheme." << endl;

cout << endl;

cout << "================================================================================";

cout << endl;

cout << " TWITTER: twitter.com/ServerProcessor" << endl;

cout << " EMAIL: nmakes@outlook.com";

cout << endl << endl;

cout << "================================================================================";

cout << endl;

cout << " RECOMMENDED MINIMUM SYSTEM REQUIREMENTS:" << endl;

cout << endl;

cout << " OS: Windows Platform" << endl;

cout << " CPU: 1.0 GHz (16bit, 32bit and 64bit supported)" << endl;

cout << " RAM: At least 1MB (requires more when handling files)" << endl;

cout << " VIDEO: Standard VGA Monitor" << endl;

cout << " STORAGE: 200 KB" << endl;

cout << " PERIPHERAL: Keyboard" << endl;

cout << endl;

cout << "================================================================================";

cout << endl << endl;

}

**SAMPLE OUTPUTS**

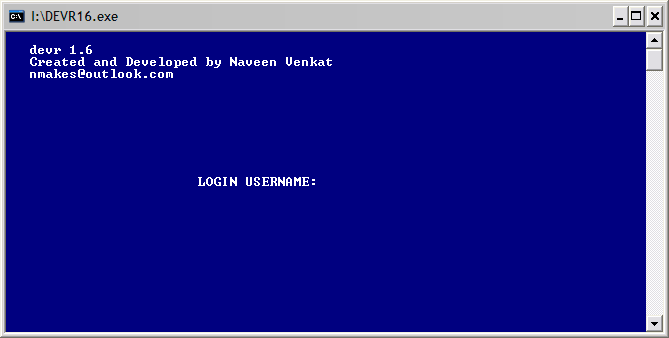


Fig. 1 ***LOGIN SCREEN***

**\_DEBUG** gives complete control to all the commands.

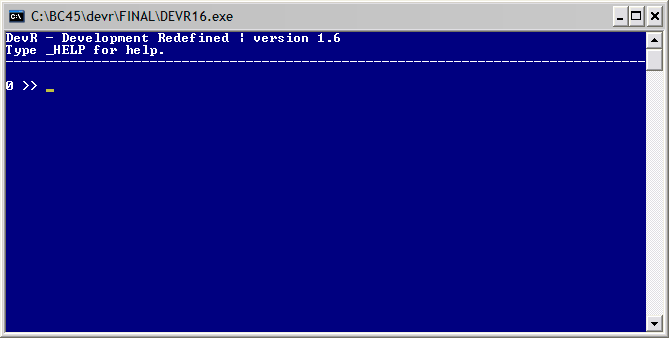


Fig. 2 ***HOME SCREEN***

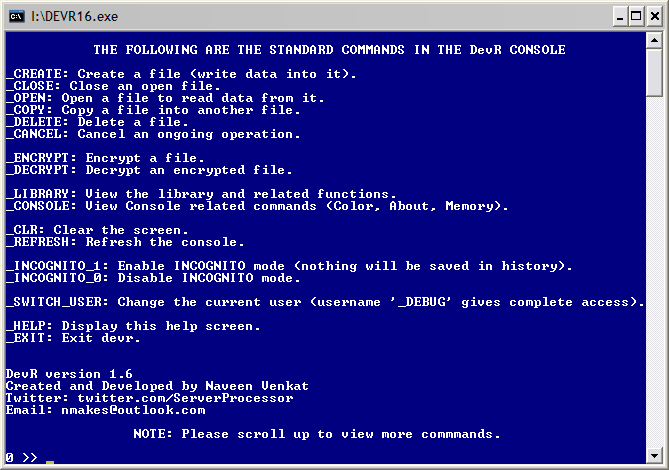


Fig. 3 ***HELP SCREEN***

(Note that the console window has been extended vertically)

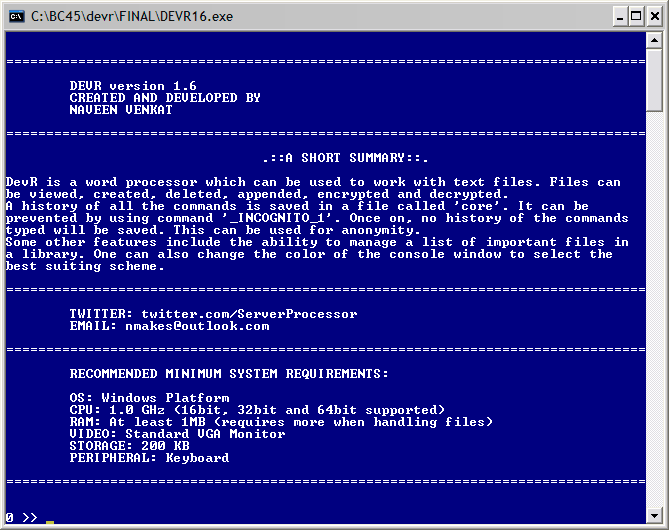


Fig 4. ***ABOUT SCREEN***

(Note that the console window has been extended vertically)

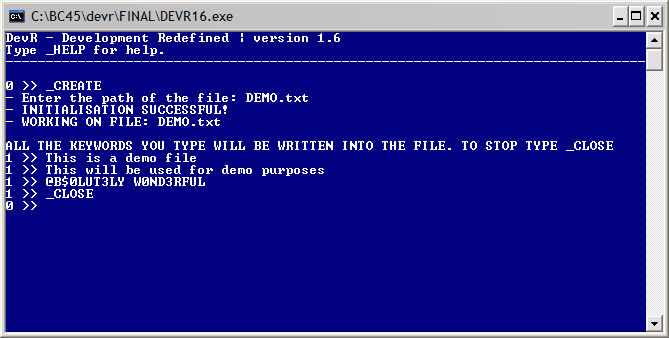


Fig 5. ***CREATING A TEXT FILE NAMED ‘DEMO.txt’***

The number 1 at the beginning of the line indicates that a file has been opened in the memory and an operation is being done on it. In this case, it is writing data into the file.

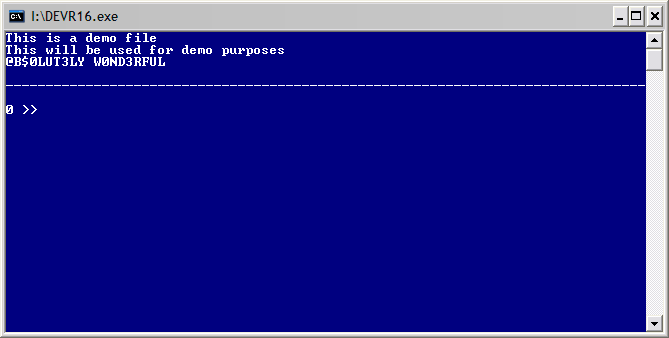


Fig. 6 ***OPENING A TEXT FILE NAMED ‘DEMO.txt’***

The program automatically prints a line right after the file ends.

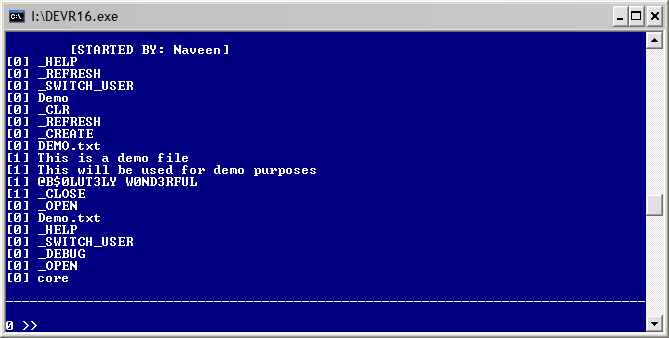


Fig. 7 ***A PORTION OF THE FILE NAMED ‘core’ WHICH SAVES A HISTORY OF ALL THE KEYWORDS TYPED***

The number 1 shows that a file was opened in the memory for an operation and 0 shows that no file was opened in the memory. At every execution, the program automatically saves the user working by writing ‘[STARTED BY: username]’ before writing the commands typed. Also note that, this file can only be opened if logged in as ‘\_DEBUG’ user.

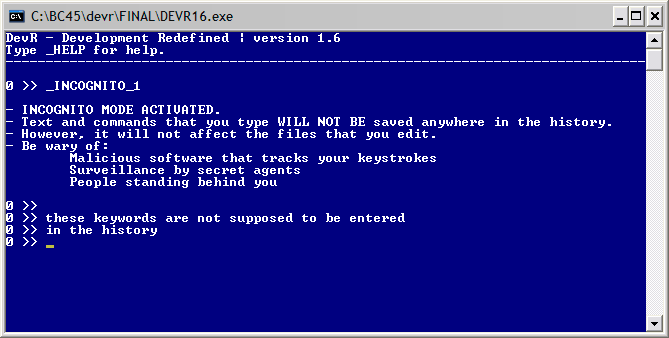


Fig. 8 ***SWITCHING INCOGNITO MODE ON. THIS INHIBITS THE SOFTWARE FROM SAVING THE KEYWORDS IN HISTORY***

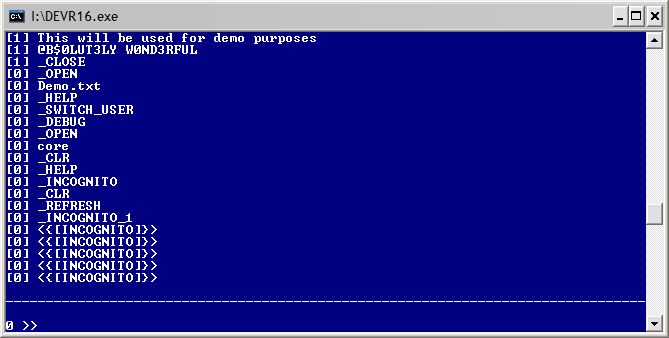


Fig. 9 ***HISTORY FILE AFTER SWITCHING ON INCOGNITO MODE***

Notice that the lines that were typed as:  
*these keywords are not supposed to be entered  
in the history*have been written as *<{[INCOGNITO]}>* in the core file. This gives privacy to the user.

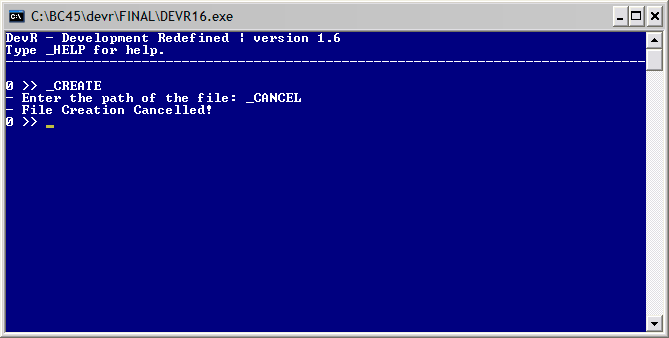


Fig. 10 ***THE USAGE OF THE COMMAND ‘\_CANCEL’ DURING THE EXECUTION OF AN EVENT***

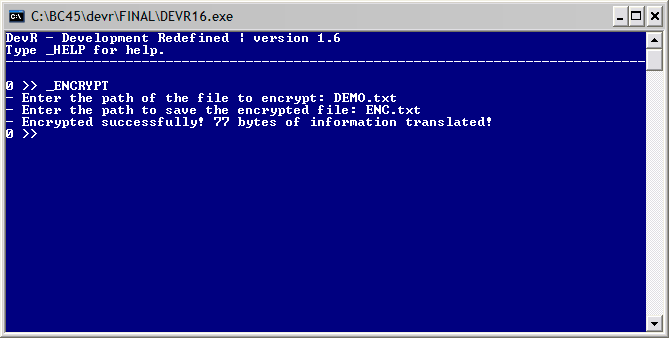


Fig. 11 ***ENCRYPTING THE TEXT FILE NAMED ‘DEMO.txt’ AND SAVING IT AS ‘ENC.txt’***

It also tells how many bytes (charecters) were translated (encrypted).

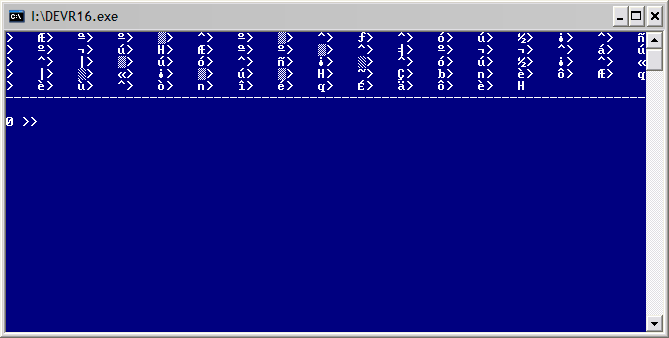


Fig. 12 ***ENCRYPTED FILE NAMED ‘ENC.txt’***

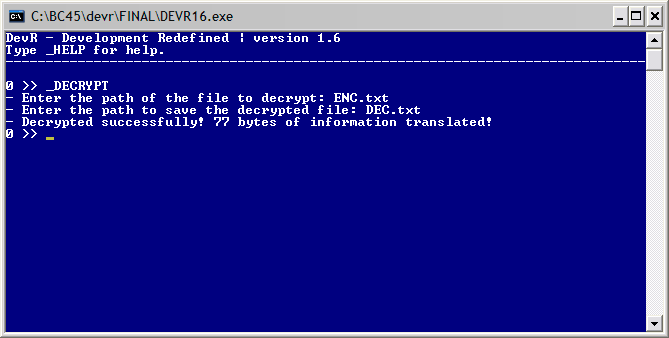


Fig. 13 ***DECRYPTING THE ENCRYPTED FILE NAMED ‘ENC.txt’ AND SAVING IT AS ‘DEC.txt’***

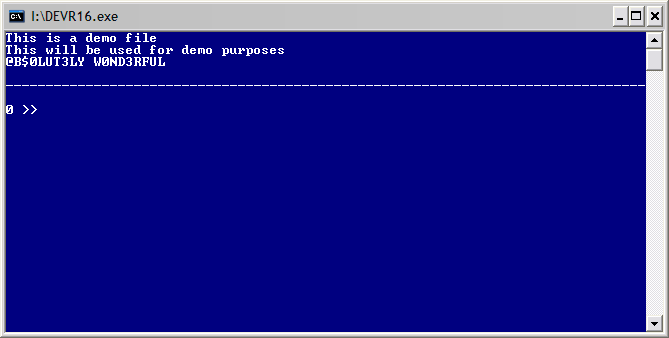


Fig. 14 ***DECRYPTED FILE.***

Notice that it is exactly identical to the original ‘DEMO.txt’ file in fig. 6

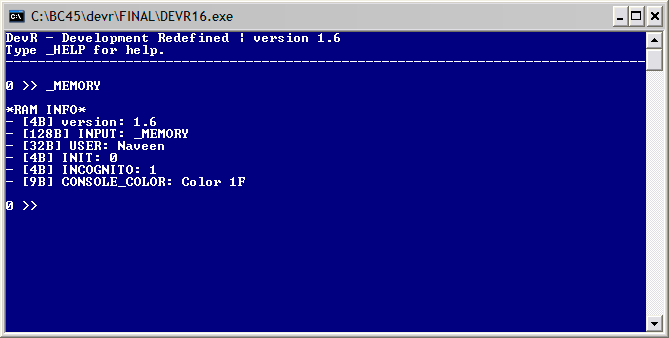


Fig. 15 ***EXECUTING THE ‘\_MEMORY’ COMMAND.***

This shows the amount of memory (in bytes) the software takes at the present state. It shows the size of some of the important variables declared in the memory. It can be used for troubleshooting.

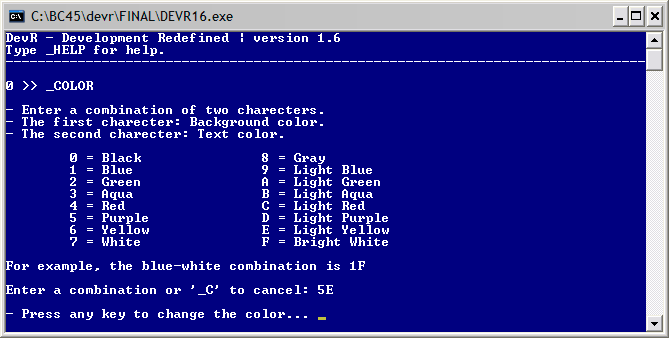


Fig. 16 ***EXECUTING THE ‘\_COLOR’ COMMAND***

It can be used to personalise the console with the most eye-pleasing color scheme.

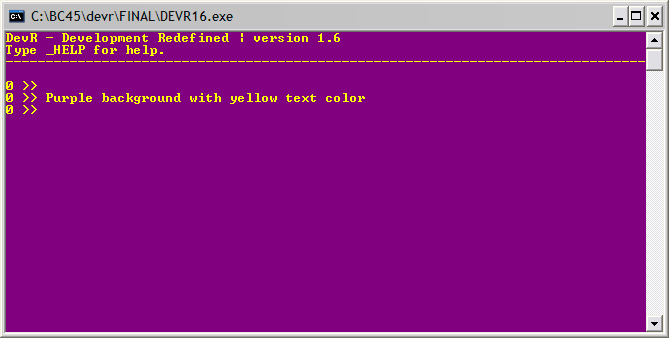


Fig. 17 ***A PURPLE – LIGHT YELLOW COLOR SCHEME***

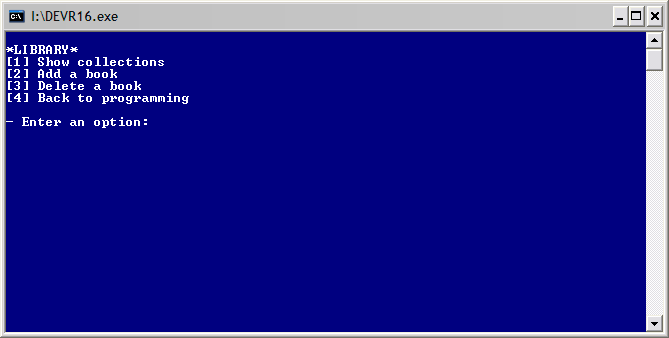


Fig. 18 ***LIBRARY***

This library can be used to list all your important files (termed as ‘books’) under one place. You can give a name to the list item and give the path where it is saved.