**29 November,2016**

Vedant Deshpande, XII Alpha,#31 CBSE Exam no.

**Blueshift**

**Nothing but a simple DBMS in C++**

**Lenovo**

bitmap1.bmp

Indira national school, wakad

Certificate

This is to certify that Master/Miss.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Of **Class XII** CBSE Exam Number:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Has prepared the project entitled ‘\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_’

Under my guidance. The report is the result of his efforts and endeavors. The report is found worthy of acceptance as final project report for the **Computer Science** subject for class XII as laid down by the CBSE Board for academic year **2016 – 2017**.

Mrs.Medha Joshi

(PGT Computer Science).

Indira National School,Wakad

Pune

Date:

Certificate

This is to certify that the project prepared by Master/Miss.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Of **Class XII** entitled ‘\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_’

for CBSE Board, All India Senior School Certificate Examination for the academic year **2016 – 2017,**for **Computer Science** subject at Indira National School, Wakad has been examined and the report is found worthy of acceptance.

\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_

Principal’s External Internal

Signature Examiner Examiner

Date: School Stamp:

**Acknowledgement**

*I would like to express my special thanks of gratitude to my teacher Medha Joshi who gave me the golden opportunity to work on BlueShift which also helped me in doing a lot of Research and thus I came to know about so many new things. Secondly I would also like to thank my parents and friends who helped me a lot in finalizing this project within the limited time frame.*

Contents

1. Working of Blueshift & OOP Concept
2. Flowchart
3. Header Files Used
4. Files Generated
5. Source Code
6. Output
7. Bibliography

Working Of BlueShift

Just like the Structured query language, Blueshift operates on databases through queries, written in the program by the end user.

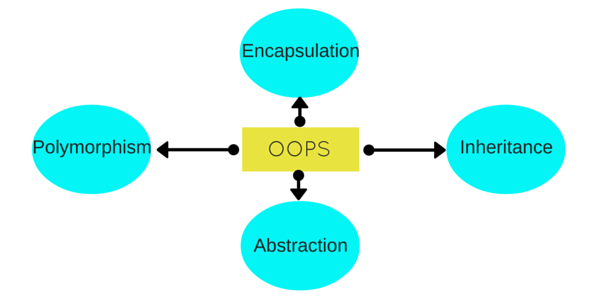
Similar to SQL, Blueshift has its own set of keywords which do the same operations on databases.

The keywords used by Blueshift are :

* new - similar to create in MySQL used to create new tables, rows, columns
* new - new col, new row
* show - similar to select in MySQL used to show data in the table, and search queries
* show - show table, show data, show avg, etc
* load - used to load table from file
* save - used to save table to file
* update - used to update data
* delete - used to delete row
* close - similar to close in MySQL closes the table, DOES NOT SAVE UNSAVED DATA

**C++ and Object Oriented Programming**

Object Oriented programming is a programming style that is associated with the concept of Class, Objects and various other concepts revolving around these two, like Inheritance, Polymorphism, Abstraction, Encapsulation etc.

* 

**Let us define what these terms mean.**

**Objects**

Objects are the basic units of OOP. They are an instance of class, which have data members and uses various member functions to perform tasks.

**Class**

It is similar to structures in C language. Class can also be defined as user defined data type also containing functions in it. So, class is basically a blueprint for object. It declares & defines what data variables the object will have and what operations can be performed on the class's object.

**Abstraction**

Abstraction refers to showing only the essential features of the application and hiding the details. In C++, classes provide methods to the outside world to access & use the data variables, but the variables are hidden from direct access. This can be done using access specifiers.

**Encapsulation**

It can also be said data binding. Encapsulation is all about binding the data variables and functions together in class.

**Inheritance**

Inheritance is a way to reuse code . The class which is inherited is called base class & the class which inherits is called derived class. So when, a derived class inherits a base class, the derived class can use all the functions which are defined in base class, hence making code reusable.

**Polymorphism** It is a feature, which lets us create functions with same name but different arguments, which will perform differently.

Source Code

#include "dynclass.h" //declaring header files

#include "authentification.h"

#include "creators\_setters.h"

#include "help.h"

#include "QandDParsers.h"

#include "updaters.h"

#include "deleter.h"

#include "file.h"

#include "resource.h"

using namespace std;

class col //Blueshift uses columns as objects and rows as array of columns

{

private:

char name[10]; //declaring attributes of column

int type;

int size;

~col();

int \*dt;

int \*dd;

char \*sd[100];

char \*sdt[100];

public:

col();

void Desc();

void nameset(char[]);

void typeset(char[]); //sets type(i.e int or char)

void typeset(int);

void sizeset(int); //sets size acc. to type

void getName(char[]); int getSize(); //getter functions

int getType();

void showdata(int);

friend void close(col \*); //can be called individually

//int column functions

void newintgenerate(int); //generates int for first row only

void intgenerate(int); //generates integer array of pointers

void intRowgenerate(int); //generates integer row

void introwinput(int);

void introwinput(int, int); //

int getIntData(int);

//char column functions

void newchargenerate(); //generates char for first row only

void chargenerate(int); //generates a character array

void charRowgenerate(int);

void charrowinput(int); //accepts input for row

void charrowinput(int, char[]);

int strcmpr(char[]);

int strcmpr(int, char[]);

void getCharData(int, char[]);};

col::col()

{

dd = NULL; //initializing pointer values

dt = NULL;

for (int i = 0; i < 100; i++) //initializing array of pointers

{

sd[i] = NULL;

sdt[i] = NULL;

}

}

void col::Desc()

{

if (type == 1)

{

cout << name << " int " << size;

}

else

{

cout << name << " char " << size;

}

}

void col::nameset(char nm[])

{

strcpy\_s(name, nm);

}

void col::typeset(char t[])

{

if (strcmp(t, "int") == 0)

{

type = 1; //int variable

}

else

{

type = 2; //string variable

}

}

void col::typeset(int t)

{

type = t;

}

void col::sizeset(int sz)

{

size = sz;

}

void col::getName(char n[])

{

//n = name;

strcpy\_s(n, 10, name);}

int col::getSize()

{

return size;

}

int col::getType()

{

return type;

}

void col::newintgenerate(int n)

{

dd = new int[n];

}

void col::newchargenerate()

{

sd[0] = new char[size];

}

void col::chargenerate(int n)

{

for (int i = 0; i < (n-1); i++)

{

sdt[i] = new char[size];

}

for (int i = 0; i < (n-1); i++)

{

strcpy\_s(sdt[i], size, sd[i]);

}

for (int i = 0; i < (n-1); i++)

{

delete[] sd[i];

}

for (int i = 0; i < n; i++)

{

sd[i] = new char[size];

}

for (int i = 0; i < (n - 1); i++)

{

strcpy\_s(sd[i], size, sdt[i]);

}

for (int i = 0; i < n; i++)

{

delete[] sdt[i];

}

}

void col::charRowgenerate(int r)

{

for (int i = 0; i < r; i++)

{

sd[i] = new char[size];

}

}

void col::intgenerate(int n)

{

dt = new int[n - 1];

for (int i = 0; i < n; i++)

{

dt[i] = dd[i];

}

delete[] dd;

dd = new int[n];

for (int i = 0; i < (n - 1); i++)

{

dd[i] = dt[i];

}

delete[] dt;

}

void col::intRowgenerate(int r)

{

dd = new int[r];

}

void col::introwinput(int n)

{

cin >> dd[n];

}

void col::introwinput(int n, int d)

{

dd[n] = d;

}

int col::getIntData(int r)

{

return dd[r];

}

void col::charrowinput(int n)

{

cin >> sd[n];

}

void col::charrowinput(int n, char s[])

{

strcpy\_s(sd[n], size, s);

}

int col::strcmpr(char qcol[])

{

return strcmp(name, qcol);

}

int col::strcmpr(int r, char qlim[])

{

return strcmp(sd[r], qlim);

}

void col::getCharData(int i, char cd[])

{

strcpy\_s(cd, size, sd[i]);

}

void col::showdata(int r)

{

if (type == 1)

{

cout << dd[r] << " ";

}

else

{

cout << sd[r] << " ";

}

}

col::~col()

{

}

void close(col \*c)

{

delete c;

}

int authenticate(); //function to authenticate at the start of the program

int authenticate()

{

char ch[10];

int i = -1;

cout << "Enter password: ";

do

{

i++;

if (ch[i] = \_getch()) //displaying \* to hide password

{

cout << '\*';

}

} while (ch[i] != ';');

ch[i] = '\0';

if (strcmp(ch, "sam") == 0) //comparing with password

{

cout << "\nCorrect password\n";

return 0;

}

else

{

cout << "\nWrong password\n";

return 1;

}

}

----------------------To Create New columns-------------------------------------

void colcreator(col \*[], int&); //function to create new column

void colNameTypeSetter(col \*[], int&); //sets column parameters

void colNameTypeSetter(col \*[], int, char[], int, int);

void newSrow(col \*[], int, int); //calls function for generating row

void newRow(col \*[], int, int);

void newRows(col \*[], int, int);

void dataEntry(col \*[], int, int); //accepts data into row

void tempshowdata(col \*[], int, int); //displays data

int tblClose(col \*[], int&);

void colcreator(col \*s[], int &n)

{

cin >> n;

for (int i = 0; i < n; i++)

{

s[i] = new col; //dynamically declaring class

}

}

void colCreator(col \*s[], int &n)

{

for (int i = 0; i < n; i++)

{

s[i] = new col;

}

cout << "\nColumns created";

}

void colNameTypeSetter(col \*s[], int& n)

{

char \*name;

char \*type;

int size;

name = new char[10];

type = new char[10];

for (int i = 0; i < n; i++)

{

cout << "BS.col>";

cin >> name >> type >> size;

s[i]->nameset(name);

s[i]->typeset(type);

s[i]->sizeset(size);

}

delete[] name;

delete[] type;

}

void colNameTypeSetter(col \*s[], int i, char nm[], int sz, int t)

{

s[i]->nameset(nm);

s[i]->typeset(t);

s[i]->sizeset(sz);

}

void newSrow(col \*s[], int n, int c)

{

for (int i = 0; i < c; i++)

{

if (s[i]->getType() == 1)

{

s[i]->newintgenerate(n);

}

else

{

s[i]->newchargenerate();

}

}

}

void newRow(col\* s[], int n, int c)

{

for (int i = 0; i < c; i++)

{

if (s[i]->getType() == 1)

{

s[i]->intgenerate(n);

}

else

{

s[i]->chargenerate(n);

}

}

}

void newRows(col\* s[], int n, int c)

{

for (int i = 0; i < c; i++)

{

if (s[i]->getType() == 1)

{

s[i]->intRowgenerate(n);

}

else

{

s[i]->charRowgenerate(n);

}

}

}

void dataEntry(col \*s[], int n, int c)

{

int ar = n - 1;

cout << "BS.row>";

for (int i = 0; i < c; i++)

{

if (s[i]->getType() == 1)

{

s[i]->introwinput(ar);

}

else

{

s[i]->charrowinput(ar);

}

}

cout << "Data entered\n";

}

void tempshowdata(col \*s[], int c, int r)

{

cout << "\n";

for (int i = 0; i < r; i++)

{

for (int j = 0; j < c; j++)

{

s[j]->showdata(i);

}

cout << "\n";

}

}

int tblClose(col \*s[], int& n)

{

for (int i = 0; i < n; i++)

{

close(s[i]);

}

cout << "Good Bye!";

return 1;

}

----------------------For Deletion from a row-------------------------------------------

void truncate(col \*[], int, int&, char[], char[]); //deletes char value

void truncate(col \*[], int, int&, char[], int); //deletes integer value

void trunc(col \*[], int, int, int);

void truncate(col \*s[], int c, int &r, char dcol[], char dval[])

{

int rows = r - 1;

for (int i = 0; i < c; i++)

{

if (s[i]->strcmpr(dcol) == 0)

{

for (int j = 0; j < rows; j++)

{

if (s[i]->strcmpr(j, dval) == 0)

{

trunc(s, c, j, rows);

r = r - 1;

return;

}

}

}

}

}

void truncate(col \*s[], int c, int &r, char dcol[], int idval)

{

int rows = r - 1;

for (int i = 0; i < c; i++)

{

if (s[i]->strcmpr(dcol) == 0)

{

for (int j = 0; j < rows; j++)

{

if (s[i]->getIntData(j) == idval)

{

trunc(s, c, j, rows);

r = r - 1;

return;

}

}

}

}

}

void trunc(col \*s[], int c, int r, int tr)

{

char data[20];

for (int i = 0; i < c; i++)

{

if (s[i]->getType() == 1)

{

for (int j = r; j < (tr - 1); j++)

{

s[i]->introwinput(j, s[i]->getIntData(j + 1));

}

}

else

{

for (int j = r; j < (tr - 1); j++)

{

s[i]->getCharData(j + 1, data);

s[i]->charrowinput(j, data);

}

}

}

}

------------------------For Saving in A File--------------------------------------------

void save(col \*[], int, int);

void load(col \*[], int&, int&);

#include <fstream>

struct table //creating structures to save the file

{

int col;

int row;

};

struct coldata //struct for table data

{

char name[10];

int type;

int size;

};

struct cdata

{

int d = 0;

char s[20];

};

void save(col \*s[], int c, int r)

{

table col;

coldata cd;

cdata d;

int rows = r - 1;

ofstream fout\_col("C:\\Users\\coldata.dat", ios::in | ios::trunc | ios::binary);

ofstream fout\_file("C:\\Users\\metadata.dat", ios::in | ios::trunc | ios::binary);

//file for table data and file for metadata is saved

if (!fout\_col) //checking for opening of file

{

cout << "FATAL ERROR: FILE NOT OPENED FOR SAVE ";

\_getch();

return;

}

fout\_col.clear();

fout\_file.clear();

col.col = c;

col.row = rows;

fout\_col.write((char\*)&col, sizeof(col)); //writing to file

for (int i = 0; i < c; i++)

{

s[i]->getName(cd.name);

cd.size = s[i]->getSize();

cd.type = s[i]->getType();

fout\_col.write((char\*)&cd, sizeof(cd));

}

fout\_col.close();

if (fout\_file.fail())

{

cout << "FATAL ERROR: FILE NOT OPENED FOR SAVE";

\_getch();

return;

}

else

{

for (int i = 0; i < c; i++)

{

if (s[i]->getType() == 1)

{

for (int j =0; j < rows; j++) //writing metadata to file

{

d.d = s[i]->getIntData(j);

fout\_file.write((char\*)&d, sizeof(d));

}

}

else

{

for (int j = 0; j < rows; j++)

{

s[i]->getCharData(j, d.s);

fout\_file.write((char\*)&d, sizeof(d));

}

}

}

}

fout\_file.close(); //closing file

cout << "File Saved!";

}

-------------------------------For loading from a file-------------------------------

void load(col \*s[], int &c, int &r)

{

table col;

coldata cd;

cdata d;

ifstream fin\_col("C:\\Users\\coldata.dat", ios::out | ios::app | ios::binary);

ifstream fin\_file("C:\\Users\\metadata.dat", ios::out | ios::binary);

//loading metadata as well as table data

if (!fin\_col || !fin\_file)

cout << "Error, files not opened";

fin\_col.read((char\*)&col, sizeof(col)); //reading data

c = col.col;

r = col.row;

colCreator(s, c); //calling column creator after loading data

for (int i = 0; i < c; i++)

{

fin\_col.read((char\*)&cd, sizeof(cd));

colNameTypeSetter(s, i, cd.name, cd.size, cd.type);

}

fin\_col.close();

newRows(s, r, c);

for (int i = 0; i < c; i++)

{

if (s[i]->getType() == 1)

{

for (int j = 0; j < r; j++)

{

fin\_file.read((char\*)&d, sizeof(d));

s[i]->introwinput(j, d.d);

}

}

else

{

for (int j = 0; j < r; j++)

{

fin\_file.read((char\*)&d, sizeof(d));

s[i]->charrowinput(j, d.s);

}

}

}

r++;

fin\_file.close();

cout << "\nfile loaded";

}

-----------------------------Help Menu---------------------------------------

void BlueShift();

void keywords();

void cmdsyntax();

void credits();

void BlueShift()

{

cout << "BlueShift is a simple, single user DMBS";

cout << " with simplified syntax which is easy to learn";

cout << "\n";

system("pause");

}

void keywords()

{

cout << "keywords of BlueShift: \n";

cout << "new - similar to create in MySQL";

cout << " used to create new tables, rows, columns";

cout << "\nkeywords of new - new col, new row";

cout << "\n";

cout << "show - similar to select in MySQL";

cout << " used to show data in the table, and search queries";

cout << "\nkeywords of show - show table, show data, show avg, etc";

cout << "load - used to load table from file";

cout << "\n";

cout << "save - used to save table to file\n";

cout << "update - used to update data\n";

cout << "delete - used to delete row\n";

cout << "close - similar to close in MySQL";

cout << " closes the table, DOES NOT SAVE UNSAVED DATA\n";

system("pause");

}

void cmdsyntax()

{

cout << "for creating columns:\n";

cout << "new col <number of columns>\n";

cout << "<col name> <col type(int/char)> <col size>";

cout << "\n";

cout << "for seeing data/doing queries:\n";

cout << "show table - for showing table properties\n ";

cout << "show data - for showing data of table\n";

cout << "show avg <col name> - shows average of column\n";

cout << "show <col name;> where <query>";

cout << "\n";

cout << "save - saving table\n";

cout << "load - loading table\n";

cout << "updating: \n";

cout << "update, and follow commands on prompt\n";

cout << "deleting: \n";

cout << "delete <col name><space><like/=><space><value of row to be deleted>\n";

system("pause");

}

void credits()

{

cout << "The Credits of making BlueShift© go to:\n";

cout << "Vedant, Vishwajeet and Chinmay";

cout << "\n";

system("pause");

}

--------------------------Searching /Updating Data in Rows------------------------------------------

void search(col \*[], int, int, char[], char[]); //searching in string type of rows

void search(col \*[], int, int, char[], int); //searching in integer rows

void updateType(col \*[], int, int); //For updating its type

void update(col \*[], int, int, char[], char[]); //For updating strings

void update(col \*[], int, int, char[], int); //For updating integers

void search(col \*s[], int c, int r, char urcol[], char urval[])

{

int rows = r - 1;

for (int i = 0; i < c; i++)

{

if (s[i]->strcmpr(urcol) == 0)

{

for (int j = 0; j < rows; j++)

{

if (s[i]->strcmpr(j, urval) == 0)

{

updateType(s, c, j);

return;

}

}

}

}

}

void search(col \*s[], int c, int r, char urcol[], int iurval)

{

int rows = r - 1;

for (int i = 0; i < c; i++)

{

if (s[i]->strcmpr(urcol) == 0)

{

for (int j = 0; j < rows; j++)

{

if (s[i]->getIntData(j) == iurval)

//searching and calling update type function

{

updateType(s, c, j);

return;

}

}

}

}

}

void updateType(col \*s[], int c, int r)

{

char \*ucol, \*uval;

char type[5];

int iuval;

ucol = new char[20];

uval = new char[20];

cout << "BS.u.what>";

cin >> ucol >> type;

if (strcmp(type, "like") == 0)

{

cin >> uval;

update(s, c, r, ucol, uval);

}

else

{

cin >> iuval;

update(s, c, r, ucol, iuval);

}

delete[] ucol;

delete[] uval;

}

void update(col \*s[], int c, int r, char ucol[], char uval[])

{

for (int i = 0; i < c; i++)

{

if (s[i]->strcmpr(ucol) == 0)

{

s[i]->charrowinput(r, uval);

}

}

}

void update(col \*s[], int c, int r, char ucol[], int iuval)

{

for (int i = 0; i < c; i++)

{

if (s[i]->strcmpr(ucol) == 0)

{

s[i]->introwinput(r, iuval);

}

}

}

--------------------------Queries----------------------------------------

void tableDesc(col \*[], int); //describing table structure

void charQueries(col \*[], int, int, char[], char[][10], int);//Calculating string queries

void intQueries(col \*[], int, int, char[], char[], char[][10], int);//Calculating integer Queries

void average(col \*[], int, int); //calculating average

void intEqual(col \*[], int, int, char[], int, char[][10], int);

void intGreaterThan(col \*[], int, int, char[], int, char[][10], int);

void intLessThan(col \*[], int, int, char[], int, char[][10], int);

void show(col \*[], int, int, char[][10], int);

//---QUERIES---

void charQueries(col \*s[], int c, int r, char qcol[], char collist[][10], int colcount)

{

char qlim[10];

cin >> qlim;

for (int i = 0; i < c; i++)

{

if (s[i]->strcmpr(qcol) == 0)

{

for (int j = 0; j < r; j++)

{

if (s[i]->strcmpr(j, qlim) == 0)

show(s, c, j, collist, colcount);

}

break;

}

}

}

void intQueries(col \*s[], int c, int r, char optor[], char qcol[], char collist[][10], int colcount)

{

int qlim;

cin >> qlim;

if (strcmp(optor, "=") == 0)

{

intEqual(s, c, r, qcol, qlim, collist, colcount);

}

if (strcmp(optor, ">") == 0)

{

intGreaterThan(s, c, r, qcol, qlim, collist, colcount);

}

if (strcmp(optor, "<") == 0)

{

intLessThan(s, c, r, qcol, qlim, collist, colcount);

}

}

void average(col \*s[], int c, int r)

{

char qcol[10];

int count = 0;

cin >> qcol;

for (int i = 0; i < c; i++)

{

if (s[i]->strcmpr(qcol) == 0)

{

for (int j = 0; j < r; j++)

{

count += s[i]->getIntData(j);

}

break;

}

}

cout << "Average " << qcol << " : " << count / r;

}

void intEqual(col \*s[], int c, int r, char qcol[], int qlim, char collist[][10], int colcount)

{

for (int i = 0; i < c; i++)

{

if (s[i]->strcmpr(qcol) == 0)

{

for (int j = 0; j < r; j++)

{

if (s[i]->getIntData(j) == qlim)

{

show(s, c, j, collist, colcount);

cout << "\n";

}

}

}

}

}

void intGreaterThan(col \*s[], int c, int r, char qcol[], int qlim, char collist[][10], int colcount)

{

for (int i = 0; i < c; i++)

{

if (s[i]->strcmpr(qcol) == 0)

{

for (int j = 0; j < r; j++)

{

if (s[i]->getIntData(j) > qlim)

{

show(s, c, j, collist, colcount);

cout << "\n";

}

}

}

}

}

void intLessThan(col \*s[], int c, int r, char qcol[], int qlim, char collist[][10], int colcount)

{

for (int i = 0; i < c; i++)

{

if (s[i]->strcmpr(qcol) == 0)

{

for (int j = 0; j < r; j++)

{

if (s[i]->getIntData(j) < qlim)

{

show(s, c, j, collist, colcount);

cout << "\n";

}

}

}

}

}

//---DISPLAY---

void tableDesc(col \*s[], int c) //describing table structure

{

cout << "\n";

for (int i = 0; i < c; i++)

{

s[i]->Desc();

cout << "\n";

}

return;

}

void show(col \*s[], int c, int r, char collist[][10], int colcount)

{

for (int i = 0; i < c; i++)

{

for (int j = 0; j < colcount; j++)

{

if (s[i]->strcmpr(collist[j]) == 0)

{

s[i]->showdata(r);

}

}

}

}

--------------------------Main Parser for Blueshift-------------------------------

int mainParser(col \*[], int&, int&);

void tableParser(col \*[], int&, int&);

void QandDParser(col \*[], int, int);

void updater(col \*[], int, int);

void deleter(col \*[], int, int&);

void help();

int mainParser(col \*s[], int& colsize, int& rowsize)

{

char keywrd[8];

cout << "\nBS.main>";

cin >> keywrd;

if (strcmp(keywrd, "new") == 0)

{

tableParser(s, colsize, rowsize); //table, data creation, input

}

if (strcmp(keywrd, "show") == 0)

{

QandDParser(s, colsize, rowsize);

}

if (strcmp(keywrd, "update") == 0)

{

updater(s, colsize, rowsize);

}

if (strcmp(keywrd, "delete") == 0)

{

deleter(s, colsize, rowsize);

}

if (strcmp(keywrd, "save") == 0)

{

save(s, colsize, rowsize);

}

if (strcmp(keywrd, "load") == 0)

{

load(s, colsize, rowsize);

}

if (strcmp(keywrd, "help") == 0)

{

help();

}

if (strcmp(keywrd, "close") == 0)

{

return tblClose(s, colsize);

}

return 0;

}

void tableParser(col \*s[], int &colsize, int& rowsize)

{

char lction[7];

cin >> lction;

if (strcmp(lction, "col") == 0)

{

colcreator(s, colsize); //column creator

colNameTypeSetter(s, colsize); //column name, type set

cout << "Columns created\n";

}

if (strcmp(lction, "row") == 0)

{

if (rowsize == 1)

{

newSrow(s, rowsize, colsize); //Generating first row

}

else

{

newRow(s, rowsize, colsize); //Generating the rest of rows

}

dataEntry(s, rowsize, colsize); //data inputting

rowsize++;

}

}

void QandDParser(col \*s[], int colsize, int rowsize)

{

char optor[5];

char lction[10], qcol[10];

char collist[10][10];

int rows = rowsize - 1, i = 0;

cin >> lction;

if (strcmp(lction, "table") == 0)

{

tableDesc(s, colsize);

return;

}

if (strcmp(lction, "data") == 0)

{

tempshowdata(s, colsize, rows);

return;

}

if (strcmp(lction, "avg") == 0)

{

average(s, colsize, rows);

return;

}

do

{

if (strcmp(lction, ";") == 0)

{

break;

}

strcpy\_s(collist[i], lction);

cin >> lction;

i++;

} while (i <= 10);

cin >> qcol >> optor;

if (strcmp(optor, "like") == 0)

{

charQueries(s, colsize, rows, qcol, collist, i);

}

else

{

intQueries(s, colsize, rows, optor, qcol, collist, i);

}

}

void updater(col \*s[], int colsize, int rowsize)

{

char \*urcol, \*urval;

char type[5];

int iurval;

urcol = new char[20];

urval = new char[20];

cout << "BS.u.ref>";

cin >> urcol >> type; //accepts column and its type to be changed

if (strcmp(type, "like") == 0)

{

cin >> urval;

search(s, colsize, rowsize, urcol, urval);

}

else

{

cin >> iurval;

search(s, colsize, rowsize, urcol, iurval);

}

delete[] urcol;

delete[] urval;

}

void deleter(col \*s[], int colsize, int &rowsize)

{

char \*dcol, type[5], \*dval;

int idval;

dcol = new char[20];

dval = new char[20];

cout << "BS.delete>where ";

cin >> dcol >> type;

if (strcmp(type, "like") == 0)

{

cin >> dval;

truncate(s, colsize, rowsize, dcol, dval);

}

else

{

cin >> idval;

truncate(s, colsize, rowsize, dcol, idval);

}

}

void help()

{

int ch;

do

{

cout << "\t\tHELP\n";

cout << "1. What is Blueshift?\n";

cout << "2. Keywords\n";

cout << "3. Command Syntax\n";

cout << "4. Exit help\n";

cout << "0. Credits\n";

cin >> ch;

switch (ch)

{

case 1:

BlueShift();

break;

case 2:

keywords();

break;

case 3:

cmdsyntax();

break;

case 4:

break;

case 0:

credits();

break;

default:

cout << "Enter proper option: ";

break;

}

} while (ch != 4);

}

-----------------------------Client Code For BlueShift----------------------------------------------

void main()

{

int colsize, rowsize = 1;

col \*s[10];

int cl = 0;

system("color 1b");

cout << "Blueshift v1.0 Alpha\n\n";

cl = authenticate();

if (cl == 1)

{

\_getch();

return;

}

cout << "Type help for help\n\n";

do

{

cl = mainParser(s, colsize, rowsize);

} while (cl != 1);

\_getch();

return;

}

Output

Blueshift v1.0 Alpha

Enter password: \*\*\*\*

Correct password

Type help for help

BS.main>help

HELP

1. What is Blueshift?

2. Keywords

3. Command Syntax

4. Exit help

0. Credits

1

BlueShift is a simple, single user DBMS with simplified syntax which is easy to learn

Press any key to continue . . .

HELP

1. What is Blueshift?

2. Keywords

3. Command Syntax

4. Exit help

0. Credits

2

keywords of BlueShift:

new - similar to create in MySQL used to create new tables, rows, columns

keywords of new - new col, new row

show - similar to select in MySQL used to show data in the table, and search queries

keywords of show - show table, show data, show avg, etc

load - used to load table from file

save - used to save table to file

update - used to update data

delete - used to delete row

close - similar to close in MySQL closes the table, DOES NOT SAVE UNSAVED DATA

Press any key to continue . . .

HELP

1. What is Blueshift?

2. Keywords

3. Command Syntax

4. Exit help

0. Credits

3

for creating columns:

new col <number of columns>

<col name> <col type(int/char)> <col size>

for seeing data/doing queries:

show table - for showing table properties

show data - for showing data of table

show avg <col name> - shows average of column

show <col name;> where <query>

save - saving table

load - loading table

updating:

update, and follow commands on prompt

deleting:

delete <col name><space><like/=><space><value of row to be deleted>

Press any key to continue . . .

HELP

1. What is Blueshift?

2. Keywords

3. Command Syntax

4. Exit help

0. Credits

0

The Credits of making BlueShift go to:

Vedant, Vishwajeet and Chinmay

1,244 lines of code!!!!

Press any key to continue . . .

BS.main>new col 2

BS.col>rollno int 2

BS.col>name char 14

Columns created

BS.main>new row

BS.row>1 Chinmay

Data entered

BS.main>new row

BS.row>2 Vedant

Data entered

BS.main>show table

rollno int 2

name char 14

BS.main>show data

1 Chinmay

2 Vedant

BS.main>update

BS.u.ref>name like vedant

BS.u.what>name like vishwajeet

BS.main>show data

1 chinmay

2 vishwajeet

BS.main>delete

BS.delete>where name like chinmay

BS.main>show data

2 vishwajeet

BS.main>new col 2

BS.col>rollno int 2

BS.col>name char 14

Columns created

BS.main>new row

BS.row>1 Chinmay

Data entered

BS.main>new row

BS.row>33 Vedant

Data entered

BS.main>show avg rollno

Average rollno : 17