ST. ANNE’S SR. SEC. SCHOOL



**SESSION 2019-20**

**COMPUTER SCIENCE**

**PROJECT REPORT**

SUBMITTED BY: Harshita Leela

CLASS: XII

SECTION: A

BOARD ROLL NO:

CERTIFICATE

This is to certify that Harshita Leela of class XII has successfully completed this project in C++ on the topic “Database Management System” prescribed by Mrs. Archana Aggarwal during the academic session 2019-20 as per the guidelines issued by the Central Board of Secondary Education. It is further certified that this is the individual work of the student.

Examiner (Internal) Examiner (External)

(Mrs. Archana Aggarwal)

ACKNOWLEDGEMENT

I would like to express my special thanks of gratitude to my teacher Mrs. Archana Aggarwal who gave me the golden opportunity to do this wonderful project on the topic “Database Management System”, which also helped me in doing a lot of research and I came to know about so many new things. I am really thankful to them.

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.

Index

1. Acknowledgement………………………2

2. Introduction……………………………...4

3. Abstract………………………………........5

4. Requirements…………………………......8

5. Help……………………………………......9

6. Input / Output Designs…………….…….12

7. Listing…………………………………….14

8. Limitations & Enhancements…………..26

INTRODUCTION

We often use various database engines but fail to know how they store the data inside  
tables or any other form.

This project helps us acknowledge how the data is stored and how it is retrieved from tables at a very basic level.

This project takes commands as input and processes them accordingly.

abstract

**Header files:**

1.fstream.h

2.conio.h

3.stdio.h

4.dos.h

5.string.h

**Functions used:**

1. char \*enter\_query();

Takes string input from user.

2. int extract\_cmd(char str[])

Extracts keywords from the query entered by the user.

Return values:

1: create

2: insert

3: search

4: delete

5: update

6: display

7: structure

8: exit

3. char \*extract\_tnm(char str[])

Returns table name entered by the user.

4. int extract\_dtype(char str[],int &i)

Returns data type extracted from the query.

Return values:

1: int (integer)

2: str (string)

5. field extract\_par(char str[],int &i)

Returns the field name and data type.

6. int extract\_val\_i(char str[],int &k)

Returns values to be entered in integer type fields.

7. void extract\_val\_s(char str[],int &i,char a[])

Returns values to be entered in string type fields.

8. int extract\_spar\_i(char str[],int &k)

Returns integer type parameter to be searched.

9. void extract\_spar\_s(char str[],int &i,char a[])

Returns string type parameter to be searched.

10. char \*extract\_s\_fnm(char str[],char s[])

Returns field name to be searched for the given value.

11. int extract\_noc(char str[])

Returns no. of fields entered by the user.

12. void create(char tnm[],char str[])

Creates the table.

13. void insert(char tnm[],char str[])

Inserts records in the table.

14. void display(char tnm[])

Displays contents of the table.

15. void display\_struct(char tnm[])

Displays structure of the table.

16. void search(char tnm[],char str[])

Searches the table for the occurrence of a record.

17. void delet(char tnm[],char str[])

Deletes the given record from the table.

18. void update(char tnm[],char str[])

Updates the given record.

Requirements

**HARDWARE REQUIREMENTS:**

1. Minimum 64 MB RAM

2. Minimum 60 MB disc space

**SOFTWARE REQUIREMENTS:**

1. Turbo C++

2. Windows 98 and above Help

**RULES:-**

1. Each keyword must start with ‘#’.

2. Each query must end with a ’;’ .

3. In case the query exceeds the screen length, enter key should be pressed and typing may be continued.

4. Each table name must start with a ‘\’.

5. Number of fields should be entered with an asterix (\*) before the number.

**DATA TYPES:-**

The supported data types are:

1. int (integer)

2. str (string)

**KEYWORDS:-**

1. #create

Syntax: #create \table\_name \*nooffields (field\_name data\_type);

This command will create a new table “table\_name” and report error if the table already exists.

2. #insert

Syntax: #insert \table\_name $value1 $value2 ;

This command will insert values in an existing table

“table\_name” and report error if the table is not found.

3. #search

Syntax: #search \table\_name @field\_name =value ;

This command will search for the occurrence of the given value of field\_name in the table “table\_name”.

4. #delete

Syntax: #delete \table\_name @field\_name =value ;

This command will delete the record with given value of field\_name in the table “table\_name”.

5. #update

Syntax: #update \table\_name @field\_name =value ;

This command will update the record with given value of field\_name in the table “table\_name”.

6. #display

Syntax: #display \table\_name ;

This command will display all the records from the table “table\_name”.

7. #structure

Syntax: #structure \table\_name ;

This command will display structure of the table “table\_name”.

8. #exit

Syntax: #exit;

This command will terminate the program.

**ERROR CODES:-**

Error code 102- This error code will be shown when the user enters invalid data types.

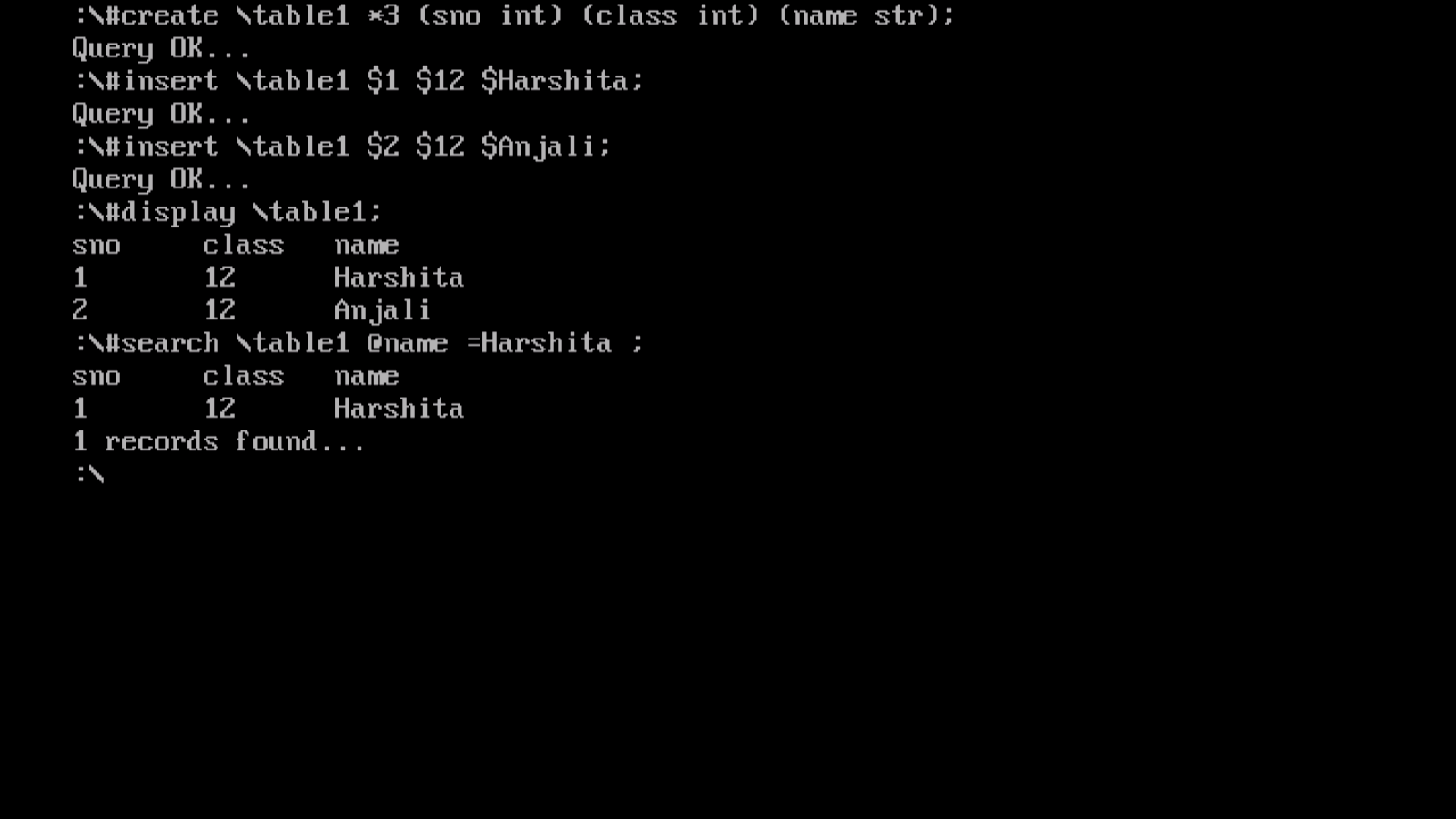
Error code 103- It means that the program was unable to open the requested table.

Error code 105- This message will be shown when the user enters a wrong keyword.

i/p-o/p designs









Listing

#include<fstream.h>

#include<conio.h>

#include<stdio.h>

#include<string.h>

#include<dos.h>

char \*enter\_query()

{ char \*str,\*query;

str=new string;

query=new char[200];

do { gets(str);

strcat(query,str);

} while(str[strlen(str)-1]!=';');

return query;

}

int extract\_cmd(char str[])

{ int i=0,j=0;

char \*s=NULL;

s=new char[20];

while(str[i++]!='#');

while(str[i]!=' ')

s[j++]=str[i++];

s[j]='\0';

if(!strcmpi(s,"create"))

return 1;

else if(!strcmpi(s,"insert"))

return 2;

else if(!strcmpi(s,"search"))

return 3;

else if(!strcmpi(s,"delete"))

return 4;

else if(!strcmpi(s,"update"))

return 5;

else if(!strcmpi(s,"display"))

return 6;

else if(!strcmpi(s,"structure"))

return 7;

else if(!strcmpi(s,"exit"))

return 8;

else

return 0;

}

char \*extract\_tnm(char str[])

{ int i=0,j=0;

char s[20];

s[0]='\0';

while(str[i++]!='\\');

while(str[i]!=' ')

s[j++]=str[i++];

s[j]='\0';

return s;

}

int extract\_dtype(char str[],int &i)

{ char \*s;

int j=0;

s=new char[20];

while(str[i]!=')')

s[j++]=str[i++];

s[j]='\0';

if(!strcmpi(s,"int"))

{ delete s;

return 1;

}

else if(!strcmpi(s,"str"))

{ delete s;

return 2;

}

else

{ delete s;

return 0;

}

}

field extract\_par(char str[],int &i)

{ char \*st;

int j=0;

field f;

st=new char[20];

while(str[i++]!='(');

while(str[i]!=' ')

st[j++]=str[i++];

i++;

st[j]='\0';

strcpy(f.fnm,st);

f.dt=extract\_dtype(str,i);

delete st;

return f;

}

int extract\_val\_i(char str[],int &k)

{ int n=0,j=0,nn,i=0;

char no[5];

while(str[k++]!='$');

while(str[k]!=' ')

no[j++]=str[k++];

no[j]='\0';

while(no[i]!='\0')

{ nn=no[i++]-48;

n=n\*10+nn;

}

return n;

}

void extract\_val\_s(char str[],int &i,char a[])

{ int j=0;

while(str[i++]!='$');

while(str[i]!=' ')

a[j++]=str[i++];

a[j]='\0';

}

int extract\_spar\_i(char str[],int &k)

{ int n=0,j=0,nn=0,i=0;

char no[5];

while(str[k++]!='=');

while(str[k]!=' ')

no[j++]=str[k++];

no[j]='\0';

while(no[i]!='\0')

{ nn=no[i++]-48;

n=n\*10+nn;

}

return n;

}

void extract\_spar\_s(char str[],int &i,char a[])

{ int j=0;

while(str[i++]!='=');

while(str[i]!=' ')

a[j++]=str[i++];

a[j]='\0';

}

char \*extract\_s\_fnm(char str[],char s[])

{ int i=0,j=0;

s[0]='\0';

while(str[i++]!='@');

while(str[i]!=' ')

s[j++]=str[i++];

s[j]='\0';

return s;

}

int extract\_noc(char str[])

{ int n=0,i=0,j=0,nn;

char no[5];

while(str[i++]!='\*');

while(str[i]!=' ')

no[j++]=str[i++];

no[j]='\0';

i=0;

while(no[i]!='\0')

{ nn=no[i++]-48;

n=n\*10+nn;

}

if(nn>10||nn<1)

return 0;

else

return n;

}

void create(char tnm[],char str[])

{ ofstream fout;

int n;

n = extract\_noc(str);

t\_struct t(n);

int i=0,j=0,k=0;

while(k++<n)

{ t.f[j++]=extract\_par(str,i);

}

if(tnm)

strcat(tnm,".dat");

fout.open(tnm,ios::binary|ios::noreplace);

if(!fout)

cout<<"Error code 103\n";

else

{ fout.write((char \*)&t,sizeof(t));

cout<<"Query OK...\n";

}

fout.close();

}

void insert(char tnm[],char str[])

{ ifstream fin;

ofstream fout;

data d;

t\_struct t;

int c=0,a=0,b=0,i=0;

if(tnm)

strcat(tnm,".dat");

fin.open(tnm,ios::binary);

if(!fin)

cout<<"Error code 103\n";

else

{ fin.seekg(0);

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

}

fin.close();

while(t.f[c].dt==1 && c<t.noc)

{ d.i[a++]=extract\_val\_i(str,i);

d.ni++;

c++;

}

while(c<t.noc)

{ extract\_val\_s(str,i,d.s[b++]);

d.ns++;

c++;

}

fout.open(tnm,ios::binary|ios::app|ios::nocreate);

if(!fout)

cout<<"Error code 103\n";

else

{

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

cout<<"Query OK...\n";

}

fout.close();

}

void display(char tnm[])

{ ifstream fin;

data d;

int i=0;

t\_struct t;

if(tnm)

strcat(tnm,".dat");

fin.open(tnm,ios::binary);

if(!fin)

cout<<"Error code 103\n";

else

{ fin.seekg(0);

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

while(i<t.noc)

{ cout<<t.f[i++].fnm<<'\t';

}

cout<<'\n';

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

{ d.disp();

}

}

fin.close();

}

void display\_struct(char tnm[])

{ ifstream fin;

int i=0;

t\_struct t;

if(tnm)

strcat(tnm,".dat");

fin.open(tnm,ios::binary);

if(!fin)

cout<<"Error code 103\n";

else

{ fin.seekg(0);

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

while(i<t.noc)

{ cout<<t.f[i-1].fnm<<'\t'<<t.f[i++].dt<<'\n';

}

}

fin.close();

}

void search(char tnm[],char str[])

{ int i=0,n=0,m=0,r=0,j=0;

char s[20],fnm[20];

ifstream fin;

t\_struct t;

data d;

extract\_s\_fnm(str,fnm);

if(tnm)

strcat(tnm,".dat");

fin.open(tnm,ios::binary);

if(!fin)

cout<<"Error code 103\n";

else

{ fin.seekg(0);

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

while(i<t.noc)

{ if(!strcmpi(t.f[i++].fnm,fnm))

break;

}

i--;

if(t.f[i].dt==1)

{ n=extract\_spar\_i(str,m);

}

else

{ extract\_spar\_s(str,m,s);

}

while(j<t.noc)

{ cout<<t.f[j++].fnm<<'\t';

}

cout<<'\n';

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

{ if(t.f[i].dt==1)

{ if(n==d.i[i])

{ d.disp();

r++;

}

}

else

{ if(!strcmpi(s,d.s[i-d.ni]))

{ d.disp();

r++;

}

}

}

}

cout<<r<<" records found...\n";

fin.close();

delete fnm;

}

void delet(char tnm[],char str[])

{ int i=0,n=0,m=0,j=0;

char s[20],fnm[20];

ifstream fin;

ofstream fout;

t\_struct t;

data d;

extract\_s\_fnm(str,fnm);

if(tnm)

strcat(tnm,".dat");

fout.open("temp.dat",ios::binary);

fin.open(tnm,ios::binary);

if(!fin||!fout)

cout<<"Error code 103\n";

else

{ fin.seekg(0);

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

while(i<t.noc)

{ if(!strcmpi(t.f[i++].fnm,fnm))

break;

}

i--;

if(t.f[i].dt==1)

{ n=extract\_spar\_i(str,m);

}

else

{ extract\_spar\_s(str,m,s);

}

cout<<"The following record(s) has been deleted: \n";

while(j<t.noc)

{ cout<<t.f[j++].fnm<<'\t';

}

cout<<'\n';

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

{ if(t.f[i].dt==1)

{ if(n==d.i[i])

{ d.disp();

}

else

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

}

else

{ if(!strcmpi(s,d.s[i-d.ni]))

{ d.disp();

}

else

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

}

}

}

fin.close();

fout.close();

remove(tnm);

rename("temp.dat",tnm);

delete fnm;

}

void update(char tnm[],char str[])

{ int i=0,n=0,j=0,m=0,c=0,k=0,l=0;

char s[20],fnm[20];

fstream finout;

t\_struct t;

data d;

extract\_s\_fnm(str,fnm);

if(tnm)

strcat(tnm,".dat");

finout.open(tnm,ios::binary|ios::in|ios::ate|ios::out);

if(!finout)

cout<<"Error code 103\n";

else

{ finout.seekg(0);

finout.read((char \*)&t,sizeof(t));

while(i<t.noc)

{ if(!strcmpi(t.f[i++].fnm,fnm))

break;

}

i--;

if(t.f[i].dt==1)

{ n=extract\_spar\_i(str,m);

}

else

{ extract\_spar\_s(str,m,s);

}

while(j<t.noc)

while(finout.read((char \*)&d,sizeof(d)))

{ if(t.f[i].dt==1)

{ if(n==d.i[i])

{ cout<<"Record to be updated:\n";

d.disp();

while(c<t.noc)

{ cout<<t.f[c].fnm<<": ";

if(t.f[c++].dt==1)

cin>>d.i[k++];

else

gets(d.s[l++]);

}

finout.seekg(finout.tellg()-sizeof(d));

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

break;

}

else

{ if(!strcmpi(s,d.s[i-d.ni]))

{ while(c<t.noc)

{ cout<<t.f[c].fnm<<": ";

if(t.f[c++].dt==1)

cin>>d.i[k++];

else

gets(d.s[l++]);

}

finout.seekg(finout.tellg()-sizeof(d));

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

break;

}

}

}

}

cout<<"Query OK...\n";

}

finout.close();

delete fnm;

}

void start()

{ clrscr();

char str[400],tname[20],ch;

int keyword=0;

do{ str[0]='\0';

tname[0]='\0';

cout<<":\\";

strcpy(str,enter\_query());

str[strlen(str)-1]='\0';

keyword=extract\_cmd(str);

strcpy(tname,extract\_tnm(str));

if(tname)

{ switch(keyword)

{ case 0: cout<<"Error code 105\n";

break;

case 1: create(tname,str);

break;

case 2: insert(tname,str);

break;

case 3: search(tname,str);

break;

case 4: delet(tname,str);

break;

case 5: update(tname,str);

break;

case 6: display(tname);

break;

case 7: display\_struct(tname);

break;

}

}

}while(keyword!=8);

cout<<"Exiting...";

delay(1000);

}

void main()

{ start();

}

Limitations

1. It offers only basic queries like create, insert, search, delete etc.

2. It has a limit of maximum 10 fields.

3. It does not offer the facility to link multiple tables in a single database, i.e. it works on just one table.

enhancements

1. Better techniques for faster traversal can be implemented.

2. It can be implemented using dynamic memory allocation.

3. More commands to perform functions like sorting of tables can be added.

bibliography

* Computer Science with C++ by Sumita Arora