**Lab Programs**

|  |  |
| --- | --- |
| **Sl.no** | **Programs** |
| 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. |
| 2 | Write a C++ program to read and write student objects with fixed-length records and the fields delimited by “|”. Implement pack ( ), unpack ( ),modify ( ) and search ( ) methods. |
| 3 | Write a C++ program to read and write student objects with Variable -Length records using any suitable record structure. Implement pack ( ),unpack ( ), modify ( ) and search ( ) methods. |
| 4 | Write a C++ program to write student objects with Variable – Length records using any suitable record structure and to read from this file a student record using RRN. |
| 5 | Write a C++ program to implement simple index on primary key for a file of student objects. Implement add ( ), search ( ), delete ( ) using the index. |
| 6 | Write a C++ program to implement index on secondary key, the name, for a file of student objects. Implement add ( ), search ( ), delete ( ) using the secondary index. |
| 7 | Write a C++ program to read two lists of names and then match the names in the two lists using Consequential Match based on a single loop. Output the names common to both the lists. |
| 8 | Write a C++ program to read k Lists of names and merge them using k way merge algorithm with k = 8. |
| 9 | Write a C++ program to implement B-Tree for a given set of integers and its operations insert ( ) and search ( ). Display the tree. |
| 10 | Write a C++ program to implement B+ tree for a given set of integers and its operations insert ( ), and search ( ). Display the tree. |
| 11 | Write a C++ program to store and retrieve student data from file using hashing. Use any collision resolution technique. |
| 12 | Write a C++ program to reclaim the free space resulting from the deletion of records using linked lists |

**Program 1**

**/\* Write a C++ program to read series of name, 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 \*/**

#include<iostream.h>

#include<stdio.h>

#include<fstream.h>

#include<conio.h>

#include<iomanip.h>

#include<stdlib.h>

**// function to reverse the string**

void reverse(char \*s,char \*r)

{

int j,len=0;

while(s[len]!='\0')

len++;

for(j=len-1;j>=0;j--)

r[len-j-1]=s[j];

r[len]='\0';

}

**// main program**

void main()

{

char name[10][20],rev[10][20],input[20],output[20],str[20],rstr[20];

int i,n,len;

fstream ifile,ofile;

clrscr();

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]);

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

reverse(name[i],rev[i]);

cout<<"the names and its reverese order are"<<endl;

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

cout<<name[i]<<setw(25)<<rev[i]<<endl;

cout<<"enter the filename which contain list of names"<<endl;

cin>>input;

ifile.open(input,ios::in);

if(!ifile)

{

cout<<"file doesnot exist";

getch();

exit(1);

}

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);

reverse(str,rstr);

ofile<<rstr<<endl;

}

getch();

}

**Output 1:**

**enter the number of names to read**

3

**enter the names**

michael j folk

bill zoellick

greg riccardi

**the names and its reverese order are**

michael j folk klof j leahcim

bill zoellick kcilleoz llib

greg riccardi idraccir gerg

**enter the filename which contain list of names**

abc.txt

**enter the filename to store reverese the names**

xyz.txt

**c:\tc>type abc.txt**

manoj kumar

praveen kollegal

vikaram narayan

sathish madappa

nemi chand

yadhu nandan

**c:\tc>type xyz.txt**

ramuk jonam

lagellok neevarp

nayaran marakiv

appadam hsihtas

dnahc imen

nadnan uhday

**Output 2:**

**enter the number of names to read**

2

**enter the names**

nagaraj poojari

shivaraj

**the names and its reverese order are**

nagaraj poojari irajoop jaragan

shivaraj jaravihs

**enter the filename which contain list of names**

pqr.txt

file doesnot exist

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

**I/O redirection : Redirect the ouput from *stdout* to a file aaa.txt**

**Syntax : program1 > filename**

**NOTE: go to command prompt**

**File- DOS Shell**

C:\tc>program name >any.txt file

**Ex:** c:\tc>prog1>aaa.txt

**c:\tc>prog1 > aaa.txt**

1

rnsit college

zzz.txt

**c:\tc>type aaa.txt**

enter the number of names to read

enter the names

the names and its reverese order are

rnsit college egelloc tisnr

enter the filename which contain list of names

file doesnot exist

**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 xyz.dat | sort**

appadam hsihtas

dnahc imen

lagellok neevarp

nadnan uhday

nayaran marakiv

ramuk jonam

**Program 2**

**/\* Write a C++ program to read and write student objects with fixed length records and the fields delimited by "|". Implement pack () and**

**unpack (), modify() and search() methods \*/**

#include<iostream.h>

#include<fstream.h>

#include<conio.h>

#include<stdio.h>

#include<iomanip.h>

#include<stdlib.h>

#include<string.h>

#define filename "std2.txt"

fstream ifile;

class student

{

char usn[15],name[20],age[5],branch[6],sem[5];

public:

void opener(fstream &ifile,char \*fn,int mode);

void read();

void pack();

void display();

void unpack();

int search();

void modify(int);

};

**// function to open a file**

void student::opener(fstream &sfile,char \*fn,int mode)

{

sfile.open(fn,mode);

if(!sfile)

{

cout<<"unable to open a file"<<endl;

getch();

exit(1);

}

}

**//function to read the student record**

void student::read()

{

cout<<"enter the usn number:"; gets(usn);

cout<<"enter the name:"; gets(name);

cout<<"enter the age:"; gets(age);

cout<<"enter the branch:"; gets(branch);

cout<<"enter the sem:"; gets(sem);

pack();

}

**// function to pack the student record using delimiter**

void student::pack()

{

char buffer[75];

strcpy(buffer,usn); strcat(buffer,"|");

strcat(buffer,name); strcat(buffer,"|");

strcat(buffer,age); strcat(buffer,"|");

strcat(buffer,branch); strcat(buffer,"|");

strcat(buffer,sem); strcat(buffer,"|");

ifile.fill('\*');

ifile<<setiosflags(ios::left)<<setw(sizeof(student))<<buffer<<endl;

}

**//function to display student record**

void student::display()

{

char buffer[75];

cout<<setiosflags(ios::left);

cout<<setw(15)<<"USN"<<setw(20)<<"NAME"<<setw(5)<<"AGE";

cout<<setw(10)<<"BRANCH"<<setw(5)<<"SEM"<<endl;

while(1)

{

unpack();

if(ifile.eof())

break;

if(usn[0]!='$')

{

cout<<setw(15)<<usn<<setw(20)<<name<<setw(5)<<age;

cout<<setw(10)<<branch<<setw(5)<<sem<<endl;

}

}

}

**// function to unpack**

void student::unpack()

{

char dummy[75];

ifile.getline(usn,15,'|');

ifile.getline(name,20,'|');

ifile.getline(age,5,'|');

ifile.getline(branch,6,'|');

ifile.getline(sem,5,'|');

ifile.getline(dummy,75,'\n');

}

**//function to search student record based on USN.**

int student::search()

{

int flag;

char susn[15];

cout<<"enter the usn to be searched:";

cin>>susn;

while(!ifile.eof())

{

flag=ifile.tellg();

unpack();

if(usn[0]!='$' && strcmp(usn,susn)==0)

{

cout<<"USN:"<<usn<<"\n"<<"NAME:"<<name<<"\n"<<"AGE:"<<age;

cout<<"\n"<<"BRANCH:"<<branch<<"\n"<<"SEM:"<<sem<<"\n";

return flag;

}

}

return -1;

}

**//function to modify record.**

void student::modify(int recpos)

{

ifile.seekp(recpos,ios::beg);

ifile.put('$');

ifile.seekp(0,ios::end);

read();

}

**//main program**

void main()

{

int ch,flag;

student s;

clrscr();

for(;;)

{

cout<<endl<<"1.- read\t2- display\t 3 .-search\t4.- modify\t5.- exit"<<endl;

cout<<"enter the choice:";

cin>>ch;

switch(ch)

{

case 1: s.opener(ifile,filename,ios::app);

cout<<"enter the student details\n";

s.read();

break;

case 2: s.opener(ifile,filename,ios::in);

cout<<"The student details are:"<<endl;

s.display();

break;

case 3:s.opener(ifile,filename,ios::in);

cout<<"Searching based on USN number"<<endl;

flag=s.search();

if(flag==-1)

cout<<"Record not found"<<endl;

break;

case 4: s.opener(ifile,filename,ios::in | ios::out | ios::nocreate);

cout<<"To modify the record based on USN"<<endl;

flag=s.search();

if(flag==-1)

cout<<"Record not found"<<endl;

else

s.modify(flag);

break;

default:

exit(0);

}

ifile.close();

}

}

**Output :**

1.- read 2- display 3 .-search 4.- modify 5.- exit

enter the choice:1

enter the student details

enter the usn number:100

enter the name:ajay

enter the age:30

enter the branch:ise

enter the sem:5

1.- read 2- display 3 .-search 4.- modify 5.- exit

enter the choice:1

enter the student details

enter the usn number:200

enter the name:suresh

enter the age:21

enter the branch:cse

enter the sem:6

1.- read 2- display 3 .-search 4.- modify 5.- exit

enter the choice:1

enter the student details

enter the usn number:300

enter the name:shashi

enter the age:20

enter the branch:me

enter the sem:2

1.- read 2- display 3 .-search 4.- modify 5.- exit

enter the choice:2

The student details are:

USN NAME AGE BRANCH SEM

100 ajay 30 ise 5

200 suresh 21 cse 6

300 shashi 20 me 2

1.- read 2- display 3 .-search 4.- modify 5.- exit

enter the choice:3

Searching based on USN number

enter the usn to be searched:200

USN:200

NAME:suresh

AGE:21

BRANCH:cse

SEM:6

1.- read 2- display 3 .-search 4.- modify 5.- exit

enter the choice:3

Searching based on USN number

enter the usn to be searched:125

Record not found

1.- read 2- display 3 .-search 4.- modify 5.- exit

enter the choice:4

To modify the record based on USN

enter the usn to be searched:300

USN:300

NAME:shashi

AGE:20

BRANCH:me

SEM:2

enter the usn number:450

enter the name:yadhu

enter the age:18

enter the branch:ece

enter the sem:2

1.- read 2- display 3 .-search 4.- modify 5.- exit

enter the choice:2

The student details are:

USN NAME AGE BRANCH SEM

100 ajay 30 ise 5

200 suresh 21 cse 6

450 yadhu 18 ece 2

1.- read 2- display 3 .-search 4.- modify 5.- exit

enter the choice:5

**c:\tc\std2.txt**

100|ajay|30|ise|5|\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

200|suresh|21|cse|6|\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

450|yadhu|18|ece|2|\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Program 3**

**/\* Write a C++ program to read and write student objects with variable-Length records using any suitable record structures. Implemet pack (), unpack (), modify () and search () methods. \*/**

#include<iostream.h>

#include<fstream.h>

#include<conio.h>

#include<stdio.h>

#include<iomanip.h>

#include<stdlib.h>

#include<string.h>

#define filename "std3.txt"

fstream ifile;

class student

{

char usn[15],name[20],age[5],branch[6],sem[5];

public:

void opener(fstream &ifile,char \*fn,int mode);

void read();

void pack();

void display();

void unpack();

int search();

void modify(int);

};

**// function to open a file**

void student::opener(fstream &sfile,char \*fn,int mode)

{

sfile.open(fn,mode);

if(!sfile)

{

cout<<"unable to open a file"<<endl;

getch();

exit(1);

}

}

**//function to read the student record**

void student::read()

{

cout<<"enter the usn number:"; gets(usn);

cout<<"enter the name:"; gets(name);

cout<<"enter the age:"; gets(age);

cout<<"enter the branch:"; gets(branch);

cout<<"enter the sem:"; gets(sem);

pack();

}

**// function to pack the student record using delimiter**

void student::pack()

{

char buffer[75];

strcpy(buffer,usn); strcat(buffer,"|");

strcat(buffer,name); strcat(buffer,"|");

strcat(buffer,age); strcat(buffer,"|");

strcat(buffer,branch); strcat(buffer,"|");

strcat(buffer,sem); strcat(buffer,"|");

ifile<<setiosflags(ios::left)<<setw(sizeof(student))<<buffer<<”#”;

}

**//function to display student record**

void student::display()

{

char buffer[75];

cout<<setiosflags(ios::left);

cout<<setw(15)<<"USN"<<setw(20)<<"NAME"<<setw(5)<<"AGE";

cout<<setw(10)<<"BRANCH"<<setw(5)<<"SEM"<<endl;

while(1)

{

unpack();

if(ifile.eof())

break;

if(usn[0]!='$')

{

cout<<setw(15)<<usn<<setw(20)<<name<<setw(5)<<age;

cout<<setw(10)<<branch<<setw(5)<<sem<<endl;

}

}

}

**// function to unpack**

void student::unpack()

{

char dummy[75];

ifile.getline(usn,15,'|');

ifile.getline(name,20,'|');

ifile.getline(age,5,'|');

ifile.getline(branch,6,'|');

ifile.getline(sem,5,'|');

ifile.getline(dummy,75,'\n');

}

**//function to search student record based on USN.**

int student::search()

{

int flag;

char susn[15];

cout<<"enter the usn to be searched:";

cin>>susn;

while(!ifile.eof())

{

flag=ifile.tellg();

unpack();

if(usn[0]!='$' && strcmp(usn,susn)==0)

{

cout<<"USN:"<<usn<<"\n"<<"NAME:"<<name<<"\n"<<"AGE:"<<age;

cout<<"\n"<<"BRANCH:"<<branch<<"\n"<<"SEM:"<<sem<<"\n";

return flag;

}

}

return -1;

}

**//function to modify record.**

void student::modify(int recpos)

{

ifile.seekp(recpos,ios::beg);

ifile.put('$');

ifile.seekp(0,ios::end);

read();

}

**//main program**

void main()

{

int ch,flag;

student s;

clrscr();

for(;;)

{

cout<<endl<<"1.- read\t2- display\t 3 .-search\t4.- modify\t5.- exit"<<endl;

cout<<"enter the choice:";

cin>>ch;

switch(ch)

{

case 1: s.opener(ifile,filename,ios::app);

cout<<"enter the student details\n";

s.read();

break;

case 2: s.opener(ifile,filename,ios::in);

cout<<"The student details are:"<<endl;

s.display();

break;

case 3:s.opener(ifile,filename,ios::in);

cout<<"Searching based on USN number"<<endl;

flag=s.search();

if(flag==-1)

cout<<"Record not found"<<endl;

break;

case 4: s.opener(ifile,filename,ios::in | ios::out | ios::nocreate);

cout<<"To modify the record based on USN"<<endl;

flag=s.search();

if(flag==-1)

cout<<"Record not found"<<endl;

else

s.modify(flag);

break;

default:

exit(0);

}

ifile.close();

}

}

**Output**

1.- read 2- display 3 .-search 4.- modify 5.- exit

enter the choice:1

enter the student details

enter the usn number:100

enter the name:amar

enter the age:20

enter the branch:ise

enter the sem:6

1.- read 2- display 3 .-search 4.- modify 5.- exit

enter the choice:1

enter the student details

enter the usn number:200

enter the name:chethan

enter the age:21

enter the branch:cse

enter the sem:7

1.- read 2- display 3 .-search 4.- modify 5.- exit

enter the choice:1

enter the student details

enter the usn number:300

enter the name:guru

enter the age:22

enter the branch:8

enter the sem:ece

1.- read 2- display 3 .-search 4.- modify 5.- exit

enter the choice:8

1.- read 2- display 3 .-search 4.- modify 5.- exit

enter the choice:1

enter the student details

enter the usn number:400

enter the name:krishna

enter the age:23

enter the branch:eee

enter the sem:6

1.- read 2- display 3 .-search 4.- modify 5.- exit

enter the choice:5

1.- read 2- display 3 .-search 4.- modify 5.- exit

enter the choice:2

The student details are:

USN NAME AGE BRANCH SEM

100 amar 20 ise 6

200 chethan 21 cse 7

300 guru 22 8 ece

400 krishna 23 eee 6

1.- read 2- display 3 .-search 4.- modify 5.- exit

enter the choice:3

Searching based on USN number

enter the usn to be searched:250

Record not found

1.- read 2- display 3 .-search 4.- modify 5.- exit

enter the choice:3

Searching based on USN number

enter the usn to be searched:200

USN:200

NAME:chethan

AGE:21

BRANCH:cse

SEM:7

1.- read 2- display 3 .-search 4.- modify 5.- exit

enter the choice:4

To modify the record based on USN

enter the usn to be searched:300

USN:300

NAME:guru

AGE:22

BRANCH:8

SEM:ece

enter the usn number:guru prasad

enter the name:20

enter the age:ece

enter the branch:7

enter the sem:2

1.- read 2- display 3 .-search 4.- modify 5.- exit

enter the choice:2

The student details are:

USN NAME AGE BRANCH SEM

100 amar 20 ise 6

200 chethan 21 cse 7

guru prasad 20 ece 7 2

400 krishna 23 eee 6

1.- read 2- display 3 .-search 4.- modify 5.- exit

enter the choice:5

c:\tc\std3.txt

100|amar|20|ise|6|#200|chethan|21|cse|7|#guru prasad|20|ece|7|2|#400|krishna|23|eee|6|#

**Program 4**

**Write a C++ program to write student objects with Variable – Length records using any suitable record structure and to read from this file a student record using RRN.**

#include<iostream.h>

#include<fstream.h>

#include<conio.h>

#include<stdio.h>

#include<iomanip.h>

#include<stdlib.h>

#include<string.h>

#define filename "std4.txt"

fstream ifile;

class student

{

char usn[15],name[20],age[5],branch[6],sem[5];

public:

void opener(fstream &ifile, char \*fn,int mode);

void read();

void pack();

void display();

void unpack();

int search();

};

**//function to open a file**

void student::opener(fstream &ifile,char \*fn,int mode)

{

sfile.open(fn,mode);

if(!sfile)

{

cout<<"unable to open a file"<<endl;

getch();

exit(1);

}

}

**//function to read the student record**

void student::read()

{

cout<<"enter the usn number:"; gets(usn);

cout<<"enter the name:"; gets(name);

cout<<"enter the age:"; gets(age);

cout<<"enter the branch:"; gets(branch);

cout<<"enter the sem:"; gets(sem);

pack();

}

// function to pack the student record using delimiter

void student::pack()

{

char buffer[75];

strcpy(buffer,usn); strcat(buffer,"|");

strcat(buffer,name); strcat(buffer,"|");

strcat(buffer,age); strcat(buffer,"|");

strcat(buffer,branch); strcat(buffer,"|");

strcat(buffer,sem); strcat(buffer,"|");

sfile<<buffer<<"#";

}

**//function to display student record**

void student::display()

{

int count=0;

cout<<setiosflags(ios::left);

cout<<setw(5)<<"RRN"<<setw(15)<<"USN"<<setw(20)<<"NAME"<<setw(5);

cout<<"AGE"<<setw(10)<<"BRANCH"<<setw(5)<<"SEM"<<endl;

while(1)

{

ifile.getline(usn,15,'|');

if(ifile.eof())

break;

unpack();

count++;

cout<<setw(5)<<count<<setw(15)<<usn<<setw(20)<<name<<setw(5)<<age;

cout<<setw(10)<<branch<<setw(5)<<sem<<endl;

}

}

**// function to unpack**

void student::unpack()

{

char dummy[75];

ifile.getline(name,20,'|');

ifile.getline(age,5,'|');

ifile.getline(branch,6,'|');

ifile.getline(sem,5,'|');

ifile.getline(dummy,75,'#');

}

**//function to search student record based on rrn.**

int student::search()

{

int rrn,count=0;

char dummy[75];

cout<<"enter the rrn to be searched:";

cin>>rrn;

cout<<"RRN:"<<rrn;

while(1)

{

ifile.getline(usn,15,'|');

if(ifile.eof())

break;

count++;

if(rrn==count)

{

cout<<"\nRecord found\n";

unpack();

cout<<"USN:"<<usn<<"\n"<<"NAME:"<<name<<"\n"<<"AGE:"<<age;

cout<<"\n"<<"BRANCH:"<<branch<<"\n"<<"SEM:"<<sem<<"\n";

return 1;

}

else

ifile.getline(dummy,100,'#');

}

return -1;

}

**// Main Program**

void main()

{

int ch,pos;

student s;

clrscr();

for(;;)

{

cout<<endl<<"1.for read\t2.for display\t3.for search\t4.for exit\n";

cout<<"Enter the choice:";

cin>>ch;

switch(ch)

{

case 1: s.opener(ifile,filename,ios::app);

cout<<"enter the student details\n";

s.read();

break;

case 2: s.opener(ifile,filename,ios::in);

cout<<"The student details are:"<<endl;

s.display();

break;

case 3:s.opener(ifile,filename,ios::in);

cout<<"To search record based on Relative record number(RRN) \n";

pos=s.search();

if(pos==-1)

cout<<"\nRRN number is out of range-Record not found \n";

break;

default:exit(0);

}

ifile.close();

}

}

**Output:**

1.for read 2.for display 3.for search 4.for exit

Enter the choice:1

enter the student details

enter the usn number:100

enter the name:manoj

enter the age:21

enter the branch:ise

enter the sem:5

1.for read 2.for display 3.for search 4.for exit

Enter the choice:1

enter the student details

enter the usn number:200

enter the name:arya

enter the age:20

enter the branch:cse

enter the sem:6

1.for read 2.for display 3.for search 4.for exit

Enter the choice:1

enter the student details

enter the usn number:300

enter the name:harsha

enter the age:19

enter the branch:me

enter the sem:6

1.for read 2.for display 3.for search 4.for exit

Enter the choice:2

The student details are:

RRN USN NAME AGE BRANCH SEM

1 100 manoj 21 ise 5

2 200 arya 20 cse 6

3 300 harsha 19 me 6

1.for read 2.for display 3.for search 4.for exit

Enter the choice:3

To search record based on Relative record number(RRN)

enter the rrn to be searched:RRN:2

Record found

USN:200

NAME:arya

AGE:20

BRANCH:cse

SEM:6

1.for read 2.for display 3.for search 4.for exit

Enter the choice:3

To search record based on Relative record number(RRN)

enter the rrn to be searched:RRN:50

RRN number is out of range-Record not found

1.for read 2.for display 3.for search 4.for exit

Enter the choice:4

**C:\tc\std4.txt**

100|manoj|21|ise|5|#200|arya|20|cse|6|#300|harsha|19|me|6|#

**Program-5**

**/\* Write a C++ program to implement simple index on primary key for a file of student objects. Implement add ( ), search ( ), delete ( ) using the index.\*/**

#include<iostream.h>

#include<fstream.h>

#include<conio.h>

#include<stdio.h>

#include<iomanip.h>

#include<stdlib.h>

#include<string.h>

#define max 10

#define datafile "student5.txt"

#define indexfile "index5.txt"

fstream stdfile, indfile;

int i,indsize;

char buffer[80];

class Student

{

char dusn[15],name[20],age[5],branch[5],sem[5];

public:

void read();

void pack();

friend int search(char\*);

void recDisp(int);

void remove(int);

void dataDisp();

void unpack();

};

class index

{

public:

char iusn[15],addr[5];

void initial();

void write();

}in,id[max];

void index::initial()

{

indfile.open(indexfile,ios::in);

if(!indfile)

{

indsize=0;

return;

}

for(indsize=0;;indsize++)

{

indfile.getline(id[indsize].iusn,15,'|');

indfile.getline(id[indsize].addr,5,'\n');

if(indfile.eof())

break;

}

indfile.close();

}

void opener(fstream &sfile,char\* fn,int mode)

{

sfile.open(fn,mode);

if(!sfile)

{

cout<<"Unable to open the file\n";

exit(1);

}

}

void index::write()

{

opener(indfile,indexfile,ios::out);

for(i=0;i<indsize;i++)

indfile<<id[i].iusn<<"|"<<id[i].addr<<"\n";

indfile.close();

}

void Student::read()

{

int k;

cout<<"Enter the usn no.\n";

gets(dusn);

if(search(dusn)>=0)

{

cout<<"usn is already present,we can't add to index file\n";

return;

}

for(i=indsize;i>0;i--)

{

if(strcmp(dusn,id[i-1].iusn)<0)

id[i]=id[i-1];

else

break;

}

opener(stdfile,datafile,ios::app);

cout<<"Enter the Name\n";

gets(name);

cout<<"Enter the age\n";

gets(age);

cout<<"Enter the branch\n";

gets(branch);

cout<<"Enter the semester\n";

gets(sem);

pack();

stdfile.seekg(0,ios::end);

k=stdfile.tellg();

stdfile<<buffer<<endl;

strcpy(id[i].iusn,dusn);

itoa(k,id[i].addr,10);

indsize++;

}

void Student::pack()

{

strcpy(buffer,dusn); strcat(buffer,"|");

strcat(buffer,name); strcat(buffer,"|");

strcat(buffer,age); strcat(buffer,"|");

strcat(buffer,branch); strcat(buffer,"|");

strcat(buffer,sem); strcat(buffer,"|");

}

int search(char\* fusn)

{

int low=0,high=indsize-1;

int mid;

while(low<=high)

{

mid=(low+high)/2;

if(strcmp(fusn,id[mid].iusn)==0)

return mid;

else if(strcmp(fusn,id[mid].iusn)>0)

low=mid+1;

else

high=mid-1;

}

return -1;

}

void Student::recDisp(int pos)

{

opener(stdfile,datafile,ios::in);

stdfile.seekg(atoi(id[pos].addr),ios::beg);

cout<<"The searched record details are:\n";

cout<<setw(16)<<"USN"<<setw(16)<<"Name"<<setw(16)<<"Age"<<setw(16)<<"Branch"<<setw(16)<<"Sem"<<endl;

unpack();

}

void Student::remove(int pos)

{

opener(stdfile,datafile,ios::in|ios::out);

stdfile.seekg(atoi(id[pos].addr),ios::beg);

stdfile.put('$');

for(i=pos;i<indsize;i++)

id[i]=id[i+1];

indsize--;

}

void Student::dataDisp()

{

cout<<setiosflags(ios::left);

cout<<setw(16)<<"USN"<<setw(16)<<"Name"<<setw(16)<<"Age" \

<<setw(16)<<"Branch"<<setw(16)<<"Sem"<<endl;

while(1)

{

unpack();

if(stdfile.eof())

break;

}

}

void Student::unpack()

{

stdfile.getline(buffer,100,'\n');

i=0;

if(buffer[i]!='$')

{

cout<<"\n";

while(buffer[i]!='\0')

{

if(buffer[i]=='|')

cout<<"\t\t";

else

cout<<buffer[i];

i++;

}

}

}

void main()

{

int ch,pos,flag;

char susn[15];

Student S;

in.initial();

clrscr();

for(;;)

{

cout<<endl<<"1.Read\n2.Display\n3.Search\n4.Delete\n5.exit\n";

cin>>ch;

switch(ch)

{

case 1: cout<<"Enter student details\n";

S.read();

in.write();

break;

case 2: opener(stdfile,datafile,ios::in);

cout<<endl<<"Student Details\n";

S.dataDisp();

cout<<endl<<"Index file details are:\n";

cout<<setw(10)<<"USN"<<setw(10)<<"Address";

for(i=0;i<indsize;i++)

{

cout<<endl<<setw(10)<<id[i].iusn<<setw(10)<<id[i].addr<<endl;

}

break;

case 3: cout<<"Enter the USN to be searched\n";

cin>>susn;

flag=search(susn);

if(flag==-1)

cout<<"Record Not found\n";

else

S.recDisp(flag);

break;

case 4: cout<<"Enter the usn no to delete from the record\n";

cin>>susn;

pos=search(susn);

if(pos==-1)

cout<<"Usn No. not found\n";

else

{

S.remove(pos);

in.write();

}

break;

default: exit(0);

}

stdfile.close();

}

}

**Output :**

1 for read, 2 for display, 3 for search, 4 for delete, 5 for exit

1

enter student details :

enter the usn number=is101

enter the name=manoj kumar

enter the age=25

enter the branch=ise

enter the semeter=6

1 for read, 2 for display, 3 for search, 4 for delete, 5 for exit

2

the student details are

usn name age branch sem

is101 manoj kumar 25 ise 6

the index file details are

usn address

is101 0

1 for read, 2 for display, 3 for search, 4 for delete, 5 for exit

1

enter student details :

enter the usn number=cs201

enter the name=vikram narayan

enter the age=35

enter the branch=cse

enter the semeter=8

1 for read, 2 for display, 3 for search, 4 for delete, 5 for exit

2

the student details are

usn name age branch sem

is101 manoj kumar 25 ise 6

cs201 vikram narayan 35 cse 8

the index file details are

usn address

cs201 29

is101 0

1 for read, 2 for display, 3 for search, 4 for delete, 5 for exit

1

enter student details :

enter the usn number=me301

enter the name=pradeep

enter the age=24

enter the branch=mec

enter the semeter=7

1 for read, 2 for display, 3 for search, 4 for delete, 5 for exit

1

enter student details :

enter the usn number=ee401

enter the name=shruthi

enter the age=20

enter the branch=eee

enter the semeter=6

1 for read, 2 for display, 3 for search, 4 for delete, 5 for exit

1

enter student details :

enter the usn number=it501

enter the name=sunitha

enter the age=it

enter the branch=it

enter the semeter=6

1 for read, 2 for display, 3 for search, 4 for delete, 5 for exit

2

the student details are

usn name age branch sem

is101 manoj kumar 25 ise 6

cs201 vikram narayan 35 cse 8

me301 pradeep 24 mec 7

ee401 shruthi 20 eee 6

it501 sunitha 21 it 6

the index file details are

usn address

cs201 29

ee401 86

is101 0

it501 111

me301 61

1 for read, 2 for display, 3 for search, 4 for delete, 5 for exit

3

enter usn number to search : me301

usn = me301

name = pradeep

age = 24

branch = mec

sem = 7

1 for read, 2 for display, 3 for search, 4 for delete, 5 for exit

3

enter usn number to search : ec250

usn number record not found for search

1 for read, 2 for display, 3 for search, 4 for delete, 5 for exit

4

enter usn number to delete the record : me301

1 for read, 2 for display, 3 for search, 4 for delete, 5 for exit

2

the student details are

usn name age branch sem

is101 manoj kumar 25 ise 6

cs201 vikram narayan 35 cse 8

ee401 shruthi 20 eee 6

it501 sunitha 21 it 6

the index file details are

usn address

cs201 29

ee401 86

is101 0

it501 111

1 for read, 2 for display, 3 for search, 4 for delete, 5 for exit

4

enter usn number to delete the record : cv105

usn number not found to delete

1 for read, 2 for display, 3 for search, 4 for delete, 5 for exit

5

C:\tc\type std5.txt

is101|manoj kumar|25|ise|6|

cs201|vikram narayan|35|cse|8|

$e301|pradeep|24|mec|7|

ee401|shruthi|20|eee|6|

it501|sunitha|21|it|6|

c:\tc\tyoe index5.txt

cs201|29

ee401|86

is101|0

it501|111

**Program 6**

/\* Write a C++ program to implement index on secondary key, the name, for a file of student objects. Implement add(), search(), delete () using the secondary index. \*/

#include<iostream.h>

#include<fstream.h>

#include<conio.h>

#include<stdio.h>

#include<iomanip.h>

#include<stdlib.h>

#include<string.h>

#define max 10

#define datafile "student6.txt"

#define indexfile "index6.txt"

#define sindexfile "sindex6.txt"

fstream stdfile, indfile,sindfile;

int i,indsize,sindsize;

char buffer[80];

class Student

{

char dusn[15],name[20],age[5],branch[5],sem[5];

public:

void read();

void pack();

friend int search(char\*);

void recDisp(int);

void remove(int);

void dataDisp();

void unpack();

};

class index

{

public:

char iusn[15],addr[5];

void initial();

void write();

}in,id[max];

class sindex

{

public:

char sname[20],susn[15];

void sinitial();

void swrite();

}sin,sid[max];

void index::initial()

{

indfile.open(indexfile,ios::in);

if(!indfile)

{

indsize=0;

return;

}

for(indsize=0;;indsize++)

{

indfile.getline(id[indsize].iusn,15,'|');

indfile.getline(id[indsize].addr,5,'\n');

if(indfile.eof())

break;

}

indfile.close();

}

void sindex::sinitial()

{

sindfile.open(sindexfile,ios::in);

if(!sindfile)

{

sindsize=0;

return;

}

for(sindsize=0;;sindsize++)

{

sindfile.getline(sid[sindsize].sname,20,'|');

sindfile.getline(sid[sindsize].susn,15,'\n');

if(sindfile.eof())

break;

}

sindfile.close();

}

void opener(fstream &sfile,char\* fn,int mode)

{

sfile.open(fn,mode);

if(!sfile)

{

cout<<"Unable to open the file\n";

exit(1);

}

}

void index::write()

{

opener(indfile,indexfile,ios::out);

for(i=0;i<indsize;i++)

indfile<<id[i].iusn<<"|"<<id[i].addr<<"\n";

indfile.close();

}

void sindex::swrite()

{

opener(sindfile,sindexfile,ios::out);

for(i=0;i<sindsize;i++)

sindfile<<sid[i].sname<<"|"<<sid[i].susn<<"\n";

sindfile.close();

}

void Student::read()

{

int k;

cout<<"Enter the usn no.\n";

gets(dusn);

if(search(dusn)>=0)

{

cout<<"usn is already present,we can't add to index file\n";

return;

}

for(i=indsize;i>0;i--)

{

if(strcmp(dusn,id[i-1].iusn)<0)

id[i]=id[i-1];

else

break;

}

opener(stdfile,datafile,ios::app);

cout<<"Enter the Name\n";

gets(name);

cout<<"Enter the age\n";

gets(age);

cout<<"Enter the branch\n";

gets(branch);

cout<<"Enter the semester\n";

gets(sem);

pack();

stdfile.seekg(0,ios::end);

k=stdfile.tellg();

stdfile<<buffer<<endl;

strcpy(id[i].iusn,dusn);

itoa(k,id[i].addr,10);

indsize++;

for(i=sindsize;i>0;i--)

{

if(strcmp(name,sid[i-1].sname)<0)

sid[i]=sid[i-1];

else if((strcmp(name,sid[i-1].sname)==0) && (strcmp(dusn,sid[i-1].susn)< 0))

sid[i]=sid[i-1];

else

break;

}

strcpy(sid[i].sname,name);

strcpy(sid[i].susn,dusn);

sindsize++;

}

void Student::pack()

{

strcpy(buffer,dusn); strcat(buffer,"|");

strcat(buffer,name); strcat(buffer,"|");

strcat(buffer,age); strcat(buffer,"|");

strcat(buffer,branch); strcat(buffer,"|");

strcat(buffer,sem); strcat(buffer,"|");

}

int search(char\* fusn)

{

int low=0,high=indsize-1;

int mid;

while(low<=high)

{

mid=(low+high)/2;

if(strcmp(fusn,id[mid].iusn)==0)

return mid;

else if(strcmp(fusn,id[mid].iusn)>0)

low=mid+1;

else

high=mid-1;

}

return -1;

}

int sec\_search()

{

int j,count=0,start=-1,pos=-1;

char skey[20],rusn[20];

cout<<"\nEnter the name to search(secondary key)\n";

gets(skey);

cout<<"The searched record deatils are:"<<endl;

for(j=0;j<sindsize;j++)

if(strcmp(skey,sid[j].sname)==0)

{

cout<<sid[j].susn<<"\t\t"<<sid[j].sname<<endl;

count++;

if(count==1)

start=j;

}

if(count>1)

{

cout<<"Enter any one usn above listed:";

cin>>rusn;

for(j=start;start<j+count;start++)

if(strcmp(sid[start].susn,rusn)==0)

{

pos=start;

return pos;

}

}

else

return start;

return pos;

}

void Student::recDisp(int pos)

{

opener(stdfile,datafile,ios::in);

stdfile.seekg(atoi(id[pos].addr),ios::beg);

cout<<"The searched record details are:"<<endl;

cout<<setiosflags(ios::left);

cout<<endl<<setw(16)<<"USN"<<setw(16)<<"Name"<<setw(16)<<"Age"<<setw(16)<<"Branch"<<setw(16)<<"Sem"<<endl;

unpack();

}

void Student::remove(int pos)

{

opener(stdfile,datafile,ios::in|ios::out);

stdfile.seekg(atoi(id[pos].addr),ios::beg);

stdfile.put('$');

for(i=pos;i<indsize;i++)

id[i]=id[i+1];

indsize--;

}

void Student::dataDisp()

{

cout<<setiosflags(ios::left);

cout<<setw(16)<<"USN"<<setw(16)<<"Name"<<setw(16)<<"Age" \

<<setw(16)<<"Branch"<<setw(16)<<"Sem"<<endl;

while(1)

{

unpack();

if(stdfile.eof())

break;

}

cout<<endl<<"The index file details are:"<<endl;

cout<<setw(10)<<"usn"<<setw(10)<<"address"<<endl;

for(i=0;i<indsize;i++)

cout<<endl<<setw(10)<<id[i].iusn<<setw(10)<<setw(10)<<id[i].addr<<endl;

cout<<endl<<"The Secondary index file details are:"<<endl;

cout<<setw(20)<<"name"<<setw(15)<<"usn"<<endl;

for(i=0;i<sindsize;i++)

cout<<endl<<setw(20)<<sid[i].sname<<setw(15)<<sid[i].susn<<endl;

}

void Student::unpack()

{

stdfile.getline(buffer,100,'\n');

i=0;

if(buffer[i]!='$')

{

cout<<"\n";

while(buffer[i]!='\0')

{

if(buffer[i]=='|')

cout<<"\t\t";

else

cout<<buffer[i];

i++;

}

}

}

void main()

{

int ch,pos,spos,flag;

char susn[15];

Student S;

in.initial();

sin.sinitial();

clrscr();

for(;;)

{

cout<<endl<<"1.Read\n2.Display\n3.Search\n4.Delete\n5.exit\n";

cin>>ch;

switch(ch)

{

case 1: cout<<"Enter student details\n";

S.read();

in.write();

sin.swrite();

break;

case 2: opener(stdfile,datafile,ios::in);

cout<<endl<<"Student Details and ";

cout<<endl<<"Index file and secondary index file details are:\n";

S.dataDisp();

break;

case 3: cout<<"To search based on secondary key\n";

spos=sec\_search();

if(spos==-1)

cout<<"Record Not found\n";

else

{

pos=search(sid[spos].susn);

S.recDisp(pos);

}

break;

case 4: spos=sec\_search();

if(spos==-1)

cout<<"no data record found\n";

else

{

cout<<"record is deleted";

pos=search(sid[spos].susn);

S.remove(pos);

for(i=spos;i<sindsize;i++)

{

sid[i]=sid[i+1];

}

sindsize--;

in.write();

sin.swrite();

}

break;

default: exit(0);

}

stdfile.close();

}

}

//Output :

1 for read, 2 for display, 3 for search, 4 for delete, 5 for exit : 1

enter student details :

enter the usn number=is101

enter the name=manoj

enter the age=32

enter the branch=ise

enter the semester=8

1 for read, 2 for display, 3 for search, 4 for delete, 5 for exit :1

enter student details :

enter the usn number=cs202

enter the name=dinesh

enter the age=23

enter the branch=cse

enter the semester=6

1 for read, 2 for display, 3 for search, 4 for delete, 5 for exit : 1

enter student details :

enter the usn number=me301

enter the name=shivraj

enter the age=22

enter the branch=mech

enter the semester=8

1 for read, 2 for display, 3 for search, 4 for delete, 5 for exit : 1

enter student details :

enter the usn number=ec404

enter the name=manoj

enter the age=30

enter the branch=ece

enter the semester=8

1 for read, 2 for display, 3 for search, 4 for delete, 5 for exit : 1

enter student details :

enter the usn number=me500

enter the name=manoj

enter the age=31

enter the branch=mech

enter the semester=6

1 for read, 2 for display, 3 for search, 4 for delete, 5 for exit : 2

the student details are

usn name age branch sem

is101 manoj 32 ise 8

cs202 dinesh 23 cse 6

me301 shivraj 22 mech 8

ec404 manoj 30 ece 8

me500 manoj 31 mech 6

the primary index file details are

usn address

cs202 23

ec404 73

is101 0

me301 47

me500 96

the secondary index file details are

name primary reference

dinesh cs202

manoj ec404

manoj is101

manoj me500

shivraj me301

1 for read, 2 for display, 3 for search, 4 for delete, 5 for exit : 3

enter the secondary key (name) to search : manoj

name=manoj usn=ec404

name=manoj usn=is101

name=manoj usn=me500

enter which usn number record to diplay : is101

usn = is101

name = manoj

age = 32

branch = ise

sem = 8

1 for read, 2 for display, 3 for search, 4 for delete, 5 for exit : 3

enter the secondary key (name) to search : shivraj

name=shivraj usn=me301

usn = me301

name = shivraj

age = 22

branch = mech

sem = 8

1 for read, 2 for display, 3 for search, 4 for delete, 5 for exit : 4

enter the secondary key (name) to search :manoj

name=manoj usn=ec404

name=manoj usn=is101

name=manoj usn=me500

enter which usn number record to diplay :me500

usn = me500

name = manoj

age = 31

branch = mech

sem = 6

1 for read, 2 for display, 3 for search, 4 for delete, 5 for exit : 2

the student details are

usn name age branch sem

is101 manoj 32 ise 8

cs202 dinesh 23 cse 6

me301 shivraj 22 mech 8

ec404 manoj 30 ece 8

the primary index file details are

usn address

cs202 23

ec404 73

is101 0

me301 47

the secondary index file details are

name primary reference

dinesh cs202

manoj ec404

manoj is101

shivraj me301

1 for read, 2 for display, 3 for search, 4 for delete, 5 for exit : 4

enter the secondary key (name) to search : shivraj

name= shivraj usn=me301

usn = me301

name = shivraj

age = 22

branch = mech

sem = 8

1 for read, 2 for display, 3 for search, 4 for delete, 5 for exit : 3

enter the secondary key (name) to search : shivraj

record not found

1 for read, 2 for display, 3 for search, 4 for delete, 5 for exit : 2

the student details are

usn name age branch sem

is101 manoj 32 ise 8

cs202 dinesh 23 cse 6

ec404 manoj 30 ece 8

the primary index file details are

usn address

cs202 23

ec404 73

is101 0

the secondary index file details are

name primary reference

dinesh cs202

manoj ec404

manoj is101

1 for read, 2 for display, 3 for search, 4 for delete, 5 for exit : 5

**c:\tc\prog6.dat**

is101|manoj|32|ise|8|

cs202|dinesh|23|cse|6|

$e301|shivraj|22|mech|8|

ec404|manoj|30|ece|8|

$e500|manoj|31|mech|6|

**c:\tc\pindex6.dat**

cs202|23

ec404|73

is101|0

**c:\tc\sindex6.dat**

dinesh|cs202

manoj|ec404

manoj|is101

**Program 7**

**/\* Write a C++ program to read two lists of names and then match the names in the two lists using consequential Match based on a single loop. Output the names common to both the files \*/**

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

#include<string.h>

#include<fstream.h>

void to\_open(fstream &file,char \*fn,int mode)

{

file.open(fn,mode);

if(!file)

{

cout<<"unable to open the file"<<endl;

getch();

exit(1);

}

}

void match(fstream &file1,fstream &file2,fstream &ofile)

{

char s1[26],s2[26];

int moreitems;

file1.getline(s1,25,'\n');

file2.getline(s2,25,'\n');

moreitems=strcmp(s1,"")&& strcmp(s2,"");

while(moreitems||(!file1.eof() && !file2.eof()))

{

if(strcmp(s1,s2)==0)

{

ofile<<s1<<"\n";

cout<<s1<<endl;

file1.getline(s1,25,'\n');

file2.getline(s2,25,'\n');

moreitems=strcmp(s1,"") && strcmp(s2,"");

}

else if(strcmp(s1,s2)<0)

{

file1.getline(s1,25,'\n');

moreitems=strcmp(s1,"");

}

else

{

file2.getline(s2,25,'\n');

moreitems=strcmp(s2,"");

}

}

}

void main()

{

fstream list1,list2,outlist;

clrscr();

to\_open(list1,"nameLst1.txt",ios::in);

to\_open(list2,"nameLst2.txt",ios::in);

to\_open(outlist,"nameLstF.txt",ios::out);

match(list1,list2,outlist);

cout<<"From namelst1.txt and namelst2.txt, matching names are in namelstF.txt"<<endl;

list1.close();

list2.close();

outlist.close();

getch();

}

**NOTE: In file name1.txt and name2.txt**

**Names should be in ascending order**

**Program 8**

**/\*Write a C++ program to read k Lists of names and merge them using k-way merge algorithm with k = 8. \*/**

#include<stdio.h>

#include<conio.h>

#include<fstream.h>

#include<string.h>

#include<stdlib.h>

#include<iostream.h>

#define K 8

void opener(fstream &file,char \*fn,int mode)

{

file.open(fn,mode);

if(!file)

{

cout<<"unable to open the file"<<endl;

getch();

exit(1);

}

}

void main()

{

fstream list[8],outfile;

char name[8][20]={"name0110.txt","name1110.txt","name2110.txt","name3110.txt","name4110.txt","name5110.txt","name6110.txt","name7110.txt"};

char item[8][20],min[20]="";

int i,count=0;

for(i=0;i<K;i++)

opener(list[i],name[i],ios::in);

opener(outfile,"merge110.txt",ios::out);

for(i=0;i<K;i++)

{

list[i].getline(item[i],20,'\n');

if(list[i].eof())

count++;

}

cout<<"Names after merging using K-Merge algorithm are:\n";

while(count<K)

{

strcpy(min,"");

for(i=0;i<K;i++)

if(strcmp(item[i],"\*")!=0)

{

strcpy(min,item[i]);

break;

}

count=0;

for(i=0;i<K;i++)

{

if(strcmp(item[i],"\*")==0)

count++;

else if(strcmp(item[i],min)<0)

strcpy(min,item[i]);

}

if(count==8)

{

break;

}

outfile<<min<<endl;

cout<<min<<endl;

for(i=0;i<K;i++)

if(strcmp(item[i],min)==0)

list[i].getline(item[i],20,'\n');

}

for(i=0;i<8;i++)

list[i].close();

getch();

}

**Program 9**

**/\* Write a C++ program to implement B-Tree for a given set of integers and its operations insert ( ) and search ( ). Display the tree. \*/**

#include<iostream.h>

#include<conio.h>

#include<stdio.h>

#include<math.h>

#include<stdlib.h>

struct node

{

int ele[4];

int child[4];

};

class btree

{

public: node\* tree[10][10];

int count[10],leaf,path[10];

btree();

node\* create\_node();

void insert(int);

void main\_search(int);

void display\_tree();

void insert\_node(node\*, int);

void search(int);

int search\_node(node\*, int);

int nodefull(node\*);

void split(node\*);

};

btree::btree()

{

leaf=-1;

for(int i=0;i<10;i++)

{

count[i]=-1;

path[i]=-1;

}

}

node\* btree::create\_node()

{

node \*n;

n=new node;

for(int i=0;i<4;i++)

{

n->ele[i]=-1;

n->child[i]=-1;

}

return n;

}

**//function to insert a key**

void btree::insert(int key)

{

int n;

if(leaf==-1)

{

tree[0][0]= create\_node();

leaf++;

count[0]++;

first\_node->ele[0]=key;

}

else if(leaf==0)

{

if(nodefull(tree[0][0]))

{

path[leaf]=0;

split(tree[0][0]);

insert(key);

}

else insert\_node(tree[0][0],key);

}

else

{

search(key);

n=path[leaf];

if(nodefull(tree[leaf][n]))

{

split(tree[leaf][n]);

insert(key);

}

else

insert\_node(tree[leaf][n],key);

}

}

void btree::main\_search(int key)

{

int flag=0,i;

node \*node1;

search(key);

node1=tree[leaf][path[leaf]];

for(i=0;node1->ele[i]!=-1;i++)

if(node1->ele[i]==key)

{

flag=1;

break;

}

cout<<"\n the path traversed is ";

for(i=0;path[i]!=-1;i++)

cout<<"->"<<path[i];

if(flag)

cout<<"\n element found \n";

else

cout<<"\n element not fount \n";

}

void btree::display\_tree()

{

int i,j,k;

for(i=0;i<=leaf;i++)

{

cout<<"\n level ---------"<<i<<"\n";

for(j=0;j<=count[i];j++)

{

for(k=0;tree[i][j]->ele[k]!=-1;k++)

cout<<" "<<tree[i][j]->ele[k];

cout<<"\t";

}

}

}

int btree::nodefull(node \*node1)

{

if(node1->ele[3]!=-1)

return 1;

else

return 0;

}

void btree::split(node \*oldnode)

{

node \*newnode,\*parent,\*n1,\*n2;

node \*temp;

int i,j,k,n,t,x,y,pos;

newnode=create\_node();

newnode->ele[0]=oldnode->ele[2];

newnode->ele[1]=oldnode->ele[3];

oldnode->ele[2]=-1;

oldnode->ele[3]=-1;

t=count[leaf];

n=path[leaf];

for(i=t,j=t+1;i>n;i--,j--)

tree[leaf][j]=tree[leaf][i];

tree[leaf][n+1]=newnode;

count[leaf]++;

x=leaf;

if(count[leaf]+1==1)

t=1;

else

t=log(count[leaf]+1)/log(2);

if(t!=leaf)

{

++leaf;

count[leaf]=count[x];

for(i=0;i<=count[leaf];i++)

{

temp=tree[leaf][i];

tree[leaf][i]=tree[x][i];

tree[x][i]=temp;

}

}

for(i=leaf-1;i>=0;i--)

count[i]=-1;

for(i=leaf,j=i-1;i>0;i--,j--)

{

for(k=0;k<(count[i]+1)/2;k++)

{

n1=tree[i][2\*k];

n2=tree[i][(2\*k)+1];

for(x=0;n1->ele[x]!=-1;x++);

for(y=0;n2->ele[y]!=-1;y++);

newnode=create\_node();

count[j]++;

tree[j][count[j]]=newnode;

newnode->ele[0]=n1->ele[x-1];

newnode->child[0]=2\*k;

newnode->ele[1]=n2->ele[y-1];

newnode->child[1]=(2\*k)+1;

}

if(count[i]!=1 && count[i]%2==0)

{

n2=tree[i][count[i]];

for(y=0;n2->ele[y]!=-1;y++);

newnode->ele[2]=n2->ele[y-1];

newnode->child[2]=count[i];

}

}

}

void btree::insert\_node(node\* node1,int key)

{

int flag=0,count=-1,i,j;

node \*newnode,\*n1;

for(i=0;i<4;i++)

if(node1->ele[i]!=-1)

++count;

i=0;

while(!flag && node1->ele[i]!=-1)

{

if(node1->ele[i]==key)

{

cout<<"Element already present";

return;

}

if(node1->ele[i]>key)

{

flag=1;

for(int j=count;j>=i;j--)

node1->ele[j+1]=node1->ele[j];

node1->ele[i]=key;

}

i++;

}

if(!flag)

{

node1->ele[count+1]=key;

for(i=leaf-1;i>=0;i--)

{

n1=tree[i][path[i]];

for(int t=0;n1->ele[t]!=-1;t++);

n1->ele[t-1]=key;

}

}

for(i=0;i<=count+1;i++)

cout<<" "<<node1->ele[i];

}

void btree::search(int key)

{

int i,j,temp;

path[0]=0;

if(leaf)

{

j=0;

for(i=0;i<leaf;i++)

{

temp=search\_node(tree[i][j],key);

path[i+1]=temp;

j=temp;

}

}

}

int btree::search\_node(node\* node1,int key)

{

for(int i=0;i<4;i++)

{

if(key<=node1->ele[i])

return node1->child[i];

else if(node1->ele[i+1]==-1)

return node1->child[i];

}

return -1;

}

int main()

{

btree bt;

int choice,key;

clrscr();

while(1)

{

cout<<"\n1.Insert\n2.Search\n3.Display\n4.Exit\nEnter your Choice\n";

cin>>choice;

switch(choice)

{

case 1:cout<<"\nEnter the element\n";

cin>>key;

bt.insert(key);

break;

case 2: cout<<"\nEnter the key\n";

cin>>key;

bt.main\_search(key);

break;

case 3: bt.display\_tree();

break;

default :exit(0);

}

}

}

**Output :**

1.insert 2.search 3.displaytree 4.exit enter your choice:1

enter the element:10

1.insert 2.search 3.displaytree 4.exit enter your choice:1

enter the element:20

10 20

1.insert 2.search 3.displaytree 4.exit enter your choice:1

enter the element:30

10 20 30

1.insert 2.search 3.displaytree 4.exit enter your choice:1

enter the element:40

10 20 30 40

1.insert 2.search 3.displaytree 4.exit enter your choice:3

level-------------0

10 20 30 40

1.insert 2.search 3.displaytree 4.exit enter your choice:1

enter the element:50

30 40 50

1.insert 2.search 3.displaytree 4.exit enter your choice:1

enter the element:60

30 40 50 60

1.insert 2.search 3.displaytree 4.exit enter your choice:1

enter the element:70

50 60 70

1.insert 2.search 3.displaytree 4.exit enter your choice:3

level-------------0

20 40 70

level-------------1

10 20 30 40 50 60 70

1.insert 2.search 3.displaytree 4.exit enter your choice:1

enter the element:25

25 30 40

1.insert 2.search 3.displaytree 4.exit enter your choice:1

enter the element:35

25 30 35 40

1.insert 2.search 3.displaytree 4.exit enter your choice:1

enter the element:37

35 37 40

1.insert 2.search 3.displaytree 4.exit enter your choice:3

level-------------0

30 70

level-------------1

20 30 40 70

level-------------2

10 20 25 30 35 37 40 50 60 70

1.insert 2.search 3.displaytree 4.exit enter your choice:2

enter the key:35

the path traversed is:->0->1->2

element found

.insert 2.search 3.displaytree 4.exit enter your choice:2

enter the key:55

the path traversed is:->0->1->2

element not found

1.insert 2.search 3.displaytree 4.exit enter your choice:1

enter the element:45

45 50 60 70

1.insert 2.search 3.displaytree 4.exit enter your choice:4

**Program 10**

**/\*Write a C++ program to implement B+ tree for a given set of integers and its operations insert ( ), and search ( ). Display the tree. \*/**

include<stdio.h>

#include<iostream.h>

#include<math.h>

#include<conio.h>

#include<stdlib.h>

struct node

{

int ele[4];

int child[4];

node \*next;

};

class bptree

{

public:node\* tree[20][20];

int count[20],leaf,path[20];

bptree();

node \*create\_node();

void insert(int);

void main\_search(int);

void display\_tree();

void insert\_node(node\*,int);

void search(int);

int search\_node(node\*,int);

int nodefull(node\*);

void split(node\*);

void seq();

};

**//constructor to initialize**

bptree::bptree()

{

leaf=-1;

for(int i=0;i<10;i++)

{

count[i]=-1;

path[i]=-1;

}

}

**//function to create a node**

node\* bptree::create\_node()

{

node \*n;

n=new node;

for(int i=0;i<4;i++)

{

n->ele[i]=-1;

n->child[i]=-1;

}

n->next=NULL;

return n;

}

void bptree::insert(int key) **//function to insert a key element**

{

int n;

if(leaf==-1)

{

tree[0][0]= create\_node();

leaf++;

count[0]++;

first\_node->ele[0]=key;

}

else if(leaf==0)

{

if(nodefull(tree[0][0]))

{

path[leaf]=0;

split(tree[0][0]);

insert(key);

}

else insert\_node(tree[0][0],key);

}

else

{

search(key);

n=path[leaf];

if((nodefull(tree[leaf][n])))

{

split(tree[leaf][n]);

insert(key);

}

else insert\_node(tree[leaf][n],key);

}

}

**//function to search for any element**

void bptree::main\_search(int key)

{

int flag=0,i;

node \*node1;

search(key);

node1=tree[leaf][path[leaf]];

for(i=0;node1->ele[i]!=-1;i++)

if(node1->ele[i]==key)

{

flag=1;

break;

}

cout<<"\nthe path traversal is:";

for(i=0;path[i]!=-1;i++)

cout<<path[i]<<"->";

if(flag)

cout<<"\nElement found\n";

else

cout<<"\nElement not found";

}

**//function to display the b+tree**

void bptree::display\_tree()

{

int i,j,k;

for(i=0;i<=leaf;i++)

{

cout<<"\nlevel-------"<<i<<"\n";

for(j=0;j<=count[i];j++)

{

if(i!=leaf)

k=1;

else

k=0;

for(;tree[i][j]->ele[k]!=-1;k++)

cout<<" "<<tree[i][j]->ele[k];

cout<<"\t";

}

}

}

**//function to display the sequence order**

void bptree::seq()

{

node\*n=tree[leaf][0];

while(n!=NULL)

{

for(int k=0;n->ele[k]!=-1;k++)

cout<<" "<<n->ele[k];

n=n->next;

}

}

**//function to search an element during insertion**

void bptree::search(int key)

{

int i,j,temp;

path[0]=0;

if(leaf)

{

j=0;

for(i=0;i<leaf;i++)

{

temp=search\_node(tree[i][j],key);

path[i+1]=temp;

j=temp;

}

}

}

**//function to search a node to insert a key**

int bptree::search\_node(node \*node1,int key)

{

for(int i=0;i<4;i++)

{

if(key<node1->ele[i])

return node1->child[i];

if((key>=node1->ele[i]) && (key<node1->ele[i+1]))

return node1->child[i];

else if(node1->ele[i+1]==-1)

return node1->child[i];

}

return -1;

}

int bptree::nodefull(node \*node1) **//function to verify the node is full or not**

{

if(node1->ele[3]!=-1)

return 1;

else

return 0;

}

**//function to insert a key in perticular posion in node**

void bptree::insert\_node(node \*node1,int key)

{

int flag=0,count=-1,i,j;

for(i=0;i<4;i++)

if(node1->ele[i]!=-1)

++count;

i=0;

while(!flag&&node1->ele[i]!=-1)

{

if(node1->ele[i]==key)

{

cout<<"element already present";

return;

}

if(node1->ele[i]>key)

{

flag=1;

for(j=count;j>=i;j--)

node1->ele[j+1]=node1->ele[j];

node1->ele[i]=key;

}

i++;

}

if(!flag)

node1->ele[count+1]=key;

for(i=0;i<=count+1;i++)

cout<<" "<<node1->ele[i];

}

**//function to split the node**

void bptree::split(node \*oldnode)

{

node \*newnode,\*n1,\*n2;

node \*temp;

int i,j,k,n,t,x;

newnode=create\_node();

newnode->ele[0]=oldnode->ele[2];

newnode->ele[1]=oldnode->ele[3];

oldnode->ele[2]=-1;

oldnode->ele[3]=-1;

t=count[leaf];

n=path[leaf];

for(i=t,j=t+1;i>n;i--,j--)

tree[leaf][j]=tree[leaf][i];

newnode->next=tree[leaf][n]->next;

tree[leaf][n]->next=newnode;

tree[leaf][n+1]=newnode;

count[leaf]++;

x=leaf;

if(count[leaf]+1==1)

t=1;

else

t=log(count[leaf]+1)/log(2);

if(t!=leaf)

{

++leaf;

count[leaf]=count[x];

for(i=0;i<=count[leaf];i++)

{

temp=tree[leaf][i];

tree[leaf][i]=tree[x][i];

tree[x][i]=temp;

}

}

for(i=leaf-1;i>=0;i--)

count[i]=-1;

for(i=t,j=i-1;i>0;i--,j--)

{

for(k=0;k<(count[i]+1)/2;k++)

{

n1=tree[i][2\*k];

n2=tree[i][(2\*k)+1];

newnode=create\_node();

count[j]++;

tree[j][count[j]]=newnode;

newnode->ele[0]=n1->ele[0];

newnode->child[0]=2\*k;

newnode->ele[1]=n2->ele[0];

newnode->child[1]=(2\*k)+1;

}

if(count[i]!=1 && count[i]%2==0)

{

n2=tree[i][count[i]];

newnode->ele[2]=n2->ele[0];

newnode->child[2]=count[i];

}

}

}

**//main program**

int main()

{

bptree bt;

int choice,key;

clrscr();

while(1)

{

cout<<"\n 1.insert\t2.search\t3.display\t4.seq\t5.exit\n enter your choice :\n";

cin>>choice;

switch(choice)

{

case 1:cout<<"\n enter the element :\n";

cin>>key;

bt.insert(key);

break;

case 2:cout<<"\n enter the key :\n";

cin>>key;

bt.main\_search(key);

break;

case 3:bt.display\_tree();

break;

case 4:bt.seq();

break;

default:exit(0);

}

}

}

**Output:**

1.insert 2.search 3.displaytree 4.seq 5 exit enter your choice:1

enter the element:10

1.insert 2.search 3.displaytree 4.seq 5 exit enter your choice:1

enter the element:20

10 20

1.insert 2.search 3.displaytree 4.seq 5 exit enter your choice:1

enter the element:30

10 20 30

1.insert 2.search 3.displaytree 4.seq 5 exit enter your choice::1

enter the element:40

10 20 30 40

1.insert 2.search 3.displaytree 4.seq 5 exit enter your choice:3

level-------------0

10 20 30 40

1.insert 2.search 3.displaytree 4.seq 5 exit enter your choice:1

enter the element:50

30 40 50

1.insert 2.search 3.displaytree 4.seq 5 exit enter your choice:1

enter the element:60

30 40 50 60

1.insert 2.search 3.displaytree 4.seq 5 exit enter your choice:1

enter the element:70

50 60 70

1.insert 2.search 3.displaytree 4.seq 5 exit enter your choice:1

enter the element:80

50 60 70 80

1.insert 2.search 3.displaytree 4.seq 5 exit enter your choice:1

enter the element:90

70 80 90

1.insert 2.search 3.displaytree 4.seq 5 exit enter your choice:3

level-------------0

50

level-------------1

30 70

level-------------2

10 20 30 40 50 60 70 80 90

1.insert 2.search 3.displaytree 4.seq 5 exit enter your choice:1

enter the element:32

30 32 40

1.insert 2.search 3.displaytree 4.seq 5 exit enter your choice:1

enter the element:34

30 32 34 40

1.insert 2.search 3.displaytree 4.seq 5 exit enter your choice:1

enter the element:36

34 36 40

1.insert 2.search 3.displaytree 4.seq 5 exit enter your choice:1

enter the element:38

34 36 38 40

1.insert 2.search 3.displaytree 4.seq 5 exit enter your choice:3

level-------------0

34

level-------------1

30 50 70

level-------------2

10 20 30 32 34 36 38 40 50 60 70 80 90

1.insert 2.search 3.displaytree 4.seq 5 exit enter your choice:4

10 20 30 32 34 36 38 40 50 60 70 80 90

1.insert 2.search 3.displaytree 4.seq 5 exit enter your choice:2

enter the key:36

the path traversed is:0->1->2->

element found

1.insert 2.search 3.displaytree 4.seq 5 exit enter your choice:2

enter the key:45

the path traversed is:0->1->3->

element not found

1.insert 2.search 3.displaytree 4.seq 5 exit enter your choice:5

**Program 11**

/\* **Write a C++ program to store and retrieve student data from file using hashing. Use any collision resolution technique.\*/**

#include<fstream.h>

#include<conio.h>

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#define studentfile "stud11.txt"

#define recsize 40

#define max 11

fstream file;

char buffer[80];

class student

{

public:

char key[15],name[20],sem[5];

void initial();

void read();

void retrieve(int addr, char k[]);

}s;

**//function to initial a file with tombstone or ####**

void student::initial()

{

int i,j;

file.open(studentfile,ios::app|ios::nocreate);

if(!file)

{

file.open(studentfile,ios::out);

if(!file)

{

cout<<"unable to open a file\n";

exit(1);

}

for(i=0;i<max;i++)

{

file.seekp(i\*recsize,ios::beg);

for(j=0;j<recsize-2;j++)

file<<"#";

file<<endl;

}

cout<<"empty file created\n";

}

file.close();

}

**//function to read and pack the student record**

void student ::read()

{

cout<<"Enter record key:"; gets(key);

cout<<"Enter the Name:"; gets(name);

cout<<"Enter Semester:"; gets(sem);

strcpy(buffer, key); strcat(buffer, "|");

strcat(buffer,name); strcat(buffer,"|");

strcat(buffer,sem); strcat(buffer,"|");

return;

}

**//hash function to generate a hask key using simple hashing algorithm**

int hash(char key[])

{

int i=0,sum=0;

while(key[i]!='\0')

{

sum=sum+key[i]-48;

i++;

}

return sum % max;

}

**// to store the record in a file using hash key**

void store(int addr)

{

char dummy[10];

int flag=0,i;

file.open(studentfile,ios::in|ios::out);

file.seekg(addr\*recsize,ios::beg);

file.getline(dummy,5,'\n');

if(strcmp(dummy,"####")==0)

{

file.seekp(addr\*recsize,ios::beg);

file<<buffer;

flag=1;

}

else

{

for(i=addr+1; i!=addr; i++)

{

if(i%max==0)

i=0;

file.seekg(i\*recsize,ios::beg);

file.getline(dummy,5,'\n');

if(strcmp(dummy,"####")==0)

{

cout<<"\n Collision has occored\n";

cout<<"Home address is "<<addr<<" actual address is "<i<<"\n";

file.seekp(i\*recsize,ios::beg);

file<<buffer;

flag=1;

break;

}

}

}

if(i==addr && (!flag))

cout<<"hash File is full, Record cannot be inserted\n";

file.close();

return;

}

**// to retrieve the student record using hasking key**

void student::retrieve(int addr, char k[])

{

int found=0,i;

char dummy[10];

i=addr;

file.open(studentfile,ios::in|ios::out);

do

{

file.seekg(i\*recsize,ios::beg);

file.getline(dummy,5,'\n');

if(strcmp(dummy,"####")==0)

break;

file.seekg(i\*recsize,ios::beg);

file.getline(key,15,'|');

if(strcmp(key,k)==0)

{

found=1;

cout<<"\nrecord found\n";

file.getline(name,20,'|');

file.getline(sem,5,'|');

cout<<"key="<<key<<"\nname="<<name<< "\nsem="<<sem;

break;

}

else

{

i++;

if(i%max==0)

i=0;

}

}while(i!=addr);

if(found==0)

cout<<"\n Record Does not exists in hash file\n";

return;

}

**//main program**

void main()

{

int ch,addr;

char skey[15];

clrscr();

s.initial();

for(;;)

{

cout<<endl<<"enter 1 for read, 2 for retrive, 3 exit\n";

cin>>ch;

switch(ch)

{

case 1:cout<<endl<<"enter student details : " <<endl;

s.read();

addr=hash(s.key);

store(addr);

break;

case 2: cout<<"\n Enter key value:";

cin>>skey;

addr=hash(skey);

s.retrieve(addr,skey);

break;

default : exit(0);

break;

}

file.close();

}

}

**Program 12**

**/\* Write a C++ program to reclaim the free space resulting from the deletion of records using linked lists.\*/**

#include<stdio.h>

#include<stdlib.h>

#include<conio.h>

#include<string.h>

#include<fstream.h>

#include<iostream.h>

class node

{

public:

char name[15], usn[15];

node \* link;

};

node \* first=NULL;

**//function to write into file**

void writefile()

{

node \* p;

char buffer[100];

fstream out;

out.open("std12.txt",ios::out);

if(!out)

{

cout<<"unable to open the file in output mode";

exit(0);

}

p=first;

while(p!=NULL)

{

strcpy(buffer,p->name);

strcat(buffer,"|");

strcat(buffer,p->usn);

strcat(buffer,"\n");

out<<buffer;

p=p->link;

}

}

**//function to display the student details**

void display()

{

node \* p;

if(first==NULL)

{

cout<<"list is empty";

return;

}

p=first;

while(p!=NULL)

{

cout<<"->"<<p->name<<"|"<<p->usn;

p=p->link;

}

}

**//function to insert the student details**

void insert()

{

char name[20], usn[15];

node \*p;

node \*q;

cout<<"enter the name : ";

cin>>name;

cout<<"enter the usn : ";

cin>>usn;

p=new node;

strcpy(p->name,name);

strcpy(p->usn,usn);

p->link=NULL;

if(first==NULL)

{

first=p;

writefile();

display();

return;

}

for(q=first;q->link!=NULL;q=q->link);

q->link=p;

writefile();

display();

}

**//function to delete the record based on usn number**

void del()

{

char usn[15];

node \* curr;

node \* prev;

node \* del;

if(first==NULL)

{

printf("the list is empty.. deletion not possible");

return;

}

cout<<"enter the usn to be deleted = ";

cin>>usn;

if(strcmp(first->usn,usn)==0)

{

cout<<"record deleted";

del=first;

first=first->link;

writefile();

delete(del);

return;

}

prev=NULL;

curr=first;

while((strcmp(curr->usn,usn)!=0) && curr!=NULL)

{

prev=curr;

curr=curr->link;

}

if(curr==NULL)

{

cout<<"the student with usn "<<usn<<" is not present";

return;

}

prev->link=curr->link;

writefile();

display();

}

**// main program**

void main()

{

int choice;

clrscr();

for(;;)

{

cout<<"\n1 : insert\_rear\t2 : delete\_id\t3 : display 4 : exit";

cout<<"\nenter the choice :";

cin>>choice;

switch(choice)

{

case 1: insert();

break;

case 2: del();

break;

case 3: display();

break;

default: exit(0);

}

}

}

**Output :**

1 : insert\_rear 2 : delete\_id 3 : display 4 : exit

enter the choice :1

enter the name : vikram

enter the usn : 200

->vikram|200

1 : insert\_rear 2 : delete\_id 3 : display 4 : exit

enter the choice :1

enter the name : sathish

enter the usn : 300

->vikram|200->sathish|300

1 : insert\_rear 2 : delete\_id 3 : display 4 : exit

enter the choice :1

enter the name : nemichand

enter the usn : 400->vikram|200->sathish|300->nemichand|400

1 : insert\_rear 2 : delete\_id 3 : display 4 : exit

enter the choice :3

->vikram|200->sathish|300->nemichand|400

1 : insert\_rear 2 : delete\_id 3 : display 4 : exit

enter the choice :2

enter the usn to be deleted = 300

->vikram|200->nemichand|400

1 : insert\_rear 2 : delete\_id 3 : display 4 : exit

enter the choice :2

enter the usn to be deleted = 500

the student with usn 500 is not present

1 : insert\_rear 2 : delete\_id 3 : display 4 : exit

enter the choice :3

->vikram|200->nemichand|400

1 : insert\_rear 2 : delete\_id 3 : display 4 : exit

enter the choice :4

**c:\tc\std12.txt**

manoj|896

raj|7845