**PROGRAM 1**

**Write a 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.**

**Source Code:**

#include<iostream>

#include<stdio.h>

#include<fstream>

#include<iomanip>

#include<stdlib.h>

using namespace std;

// function to reverse the string

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

{

int j,len;

for(len=0;s[len]!='\0';len++); // to calculate the length of string

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

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

r[len]='\0';

}

// main program

int main()

{

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

int i,n,len;

fstream ifile,ofile;

cout<<"enter the number of names to read "<<endl;

cin>>n;

cout<<"enter the names"<<endl;

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

cin>>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";

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;

}

}

**PROGRAM 2**

**Write a program to read and write student objects with fixed-length records and the fields delimited by “|”. Implement pack ( ), unpack ( ), modify ( ) and search ( ) methods.**

**Source Code:**

#include<iostream>

#include<fstream>

#include<stdlib.h>

#include<string.h>

#include<stdio.h>

#define SIZE 50

using namespace std;

fstream file;

class fixedf

{

public:

char usn[11],name[15],sem[6],dept[6];

public: void pack();

void unpack();

void search();

void modify();

};

void fixedf::pack()

{

char b[SIZE+1];

fixedf f;

cout<<"\nEnter usn, name, sem, dept: ";

cin>>f.usn>>f.name>>f.sem>>f.dept;

file.open("student.txt",ios::app);

sprintf(b,"%s|%s|%s|%s|",f.usn,f.name,f.sem,f.dept);

int len=strlen(b);

while(len<(SIZE))

{

strcat(b,"-");

len++;

}

strcat(b,"$");

file<<b;

file.close();

}

void fixedf::search()

{

char b[SIZE+1], usn[11];

fixedf f;

file.open("student.txt",ios::in);

cout<<"\nEnter usn to be searched: ";

cin>>usn;

while(!file.eof())

{

file.getline(b,52,'$');

sscanf(b,"%[^|]|%[^|]|%[^|]|%[^|]|",f.usn,f.name,f.sem,f.dept);

if(strcmp(f.usn,usn)==0)

{

cout<<"\nRecord found\n";

cout<<f.usn<<" "<<f.name<<" "<<f.sem<<" "<<f.dept<<endl;

return;

}

}

cout<<"\n Record not found";

return;

}

void fixedf::modify()

{ char b[SIZE+1],usn[11];

fixedf f;

int n=0, ch;

file.open("student.txt",ios::in|ios::out);

cout<<"\nEnter usn to be modified: ";

cin>>usn;

while(!file.eof())

{

file.getline(b,52,'$');

sscanf(b,"%[^|]|%[^|]|%[^|]|%[^|]|",f.usn,f.name,f.sem,f.dept);

if(strcmp(f.usn,usn)==0)

{

cout<<"\nKey found\n1:Modify name 2:modify sem 3: MOdify Dept\nENter your choice: ";

cin>>ch;

switch(ch)

{

case 1: cout<<"\nENter name: "; cin>>f.name; break;

case 2: cout<<"\nENter sem: "; cin>>f.sem; break;

case 3: cout<<"\nEnter dept: "; cin>>f.dept; break;

default: break;

}

sprintf(b,"%s|%s|%s|%s|",f.usn,f.name,f.sem,f.dept);

int len=strlen(b);

while(len<(SIZE))

{

strcat(b,"-");

len++;

}

strcat(b,"$");

file.seekp((n\*(SIZE+1)),ios::beg);

file<<b;

return;

}

n++;

}

cout<<"\n Record not found";

return;

}

void fixedf::unpack()

{

char b[SIZE+1];

fixedf f;

file.open("student.txt",ios::in);

while(!file.eof())

{

file.getline(b,52,'$');

if(file.eof()) break;

sscanf(b,"%[^|]|%[^|]|%[^|]|%[^|]",f.usn,f.name,f.sem,f.dept);

cout<<f.usn<<" "<<f.name<<" "<<f.sem<<" "<<f.dept<<endl;

}

}

void main()

{

fixedf f;

int ch;

for(;;)

{

cout<<"\n1: pack 2: unpack 3: search 4: Modify 5:Exit\nEnter your choice: ";

cin>>ch;

switch(ch)

{

case 1: f.pack();

break;

case 2: f.unpack();

file.close();

break;

case 3: f.search();

file.close();

break;

case 4: f.modify();

file.close();

break;

default: exit(0);

}

}

}

**PROGRAM 3**

**Write a program to read and write student objects with Variable - Length records using any suitable record structure. Implement pack ( ), unpack ( ), modify ( ) and search ( ) methods.**

**Source Code:**

#include<iostream>

#include<fstream>

#include<stdlib.h>

#include<string.h>

#include<stdio.h>

Using namespace std;

fstream file;

class variabler

{

public:

char usn[11],name[15],sem[6],dept[6];

public: void pack();

void unpack();

void search();

void modify();

};

void variabler::pack()

{

char b[100];

variabler r;

cout<<"\nEnter usn, name, sem, dept: ";

cin>>r.usn>>r.name>>r.sem>>r.dept;

file.open("s.txt",ios::app);

sprintf(b,"%s|%s|%s|%s|",r.usn,r.name,r.sem,r.dept);

strcat(b,"$");

file<<b;

file.close();

}

void variabler::search()

{

char b[100], usn[11];

int flag=0;

variabler r;

file.open("s.txt",ios::in);

cout<<"\nEnter usn to be searched: ";

cin>>usn;

while(!file.eof())

{

file.getline(b,100,'$');

sscanf(b,"%[^|]|%[^|]|%[^|]|%[^|]|",r.usn,r.name,r.sem,r.dept);

if(strcmp(r.usn,usn)==0)

{

cout<<"\nRecord found\n";

cout<<r.usn<<" "<<r.name<<" "<<r.sem<<" "<<r.dept<<endl;

flag=1;

break;

}

}

if(flag==0)

cout<<"\n Record not found";

//file.close();

}

void variabler::modify()

{ char b[100],usn[11];

variabler r[100];

int i=0,flag=0,ch;

file.open("s.txt",ios::in);

while(!file.eof())

{

file.getline(b,100,'$');

if(file.eof())

break;

sscanf(b,"%[^|]|%[^|]|%[^|]|%[^|]|",r[i].usn,r[i].name,r[i].sem,r[i].dept);

i++;

}

file.close();

cout<<"\nEnter usn to be modified: ";

cin>>usn;

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

{

if(strcmp(r[j].usn,usn)==0)

{

cout<<"\nKey found\n1:Modify name 2:modify sem 3: MOdify Dept\nENter your choice: ";

cin>>ch;

switch(ch)

{

case 1: cout<<"\nENter name: "; cin>>r[j].name; break;

case 2: cout<<"\nENter sem: "; cin>>r[j].sem; break;

case 3: cout<<"\nEnter dept: "; cin>>r[j].dept; break;

default: break;

}

flag=1;

}

}

if(flag==0)

cout<<"Record not found\n";

else

{

file.open("s.txt",ios::out);

i=0;

while(i<j)

{

sprintf(b,"%s|%s|%s|%s|",r[i].usn,r[i].name,r[i].sem,r[i].dept);

strcat(b,"$");

file<<b;

i++;

}

}

// file.close();

}

void variabler::unpack()

{

char b[100];

variabler r;

file.open("s.txt",ios::in);

while(1)

{

file.getline(b,100,'$');

if(file.eof())

break;

sscanf(b,"%[^|]|%[^|]|%[^|]|%[^|]",r.usn,r.name,r.sem,r.dept);

cout<<r.usn<<" "<<r.name<<" "<<r.sem<<" "<<r.dept<<endl;

}

//file.close();

}

void main()

{

variabler r;

int ch;

for(;;)

{

cout<<"\n1: pack 2: unpack 3: search 4: Modify 5:Exit\nEnter your choice: ";

cin>>ch;

switch(ch)

{

case 1: r.pack();

break;

case 2: r.unpack();

file.close();

break;

case 3: r.search();

file.close();

break;

case 4: r.modify();

file.close();

break;

default: exit(0);

}

}

}

**PROGRAM 4**

**Write a 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.**

**Source Code:**

#include<iostream>

#include<fstream>

#include<stdlib.h>

#include<string.h>

#include<stdio.h>

using namespace std;

fstream file;

class variable

{

public:

char usn[20],name[20],sem[2],dept[20];

public: void pack();

void unpack();

void search();

};

void variable::pack()

{

int len1,l,count = 100;

char b[100],c[150],len[10],temp[10];

variable f;

cout<<"\nEnter usn, name, sem, dept: ";

cin>>f.usn>>f.name>>f.sem>>f.dept;

file.open("stu4.txt",ios::in);

while(!file.eof())

{

file.getline(b,100,'$');

if(file.eof())

break;

sscanf(b,"|%[^|]|%[^|0]",temp,len);

len1=atoi(len);

file.seekg(len1,ios::cur);

count++;

}

file.close();

file.open("stu4.txt",ios::app);

sprintf(b,"%s|%s|%s|%s|#",f.usn,f.name,f.sem,f.dept);

l = strlen(b);

sprintf(c,"|%d|%d$",count,l);

strcat(c,b);

file.write(c,strlen(c));

file.close();

}

void variable::search()

{

char b[100],rrn[11],temp[100],len[10];

int flag=0,len1;

variable f;

file.open("stu4.txt",ios::in);

cout<<"\nEnter rrn to be searched: ";

cin>>rrn;

while(1)

{

file.getline(b,100,'$');

if(file.eof())

break;

sscanf(b,"|%[^|]|%[^|0]",temp,len);

len1=atoi(len);

if(strcmp(temp,rrn)!=0)

{

file.seekg(len1,ios::cur);

continue;

}

if(strcmp(temp,rrn)==0)

{

flag=1;

file.getline(b,100,'#');

sscanf(b,"%[^|]|%[^|]|%[^|]|%[^|]",f.usn,f.name,f.sem,f.dept);

cout<<f.usn<<" "<<f.name<<" "<<f.sem<<" "<<f.dept<<endl;

}

}

if(flag==0)

cout<<"\nKey not found";

file.close();

}

int main()

{

variable f;

int ch;

for(;;)

{

cout<<"\n1: Write Record 2: search \nEnter your choice: ";

cin>>ch;

switch(ch)

{

case 1: f.pack();break;

case 2: f.search(); break;

default: exit(0);

}

}

}

**PROGRAM 5**

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

**Source Code:**

#include<iostream>

#include<stdio.h>

#include<string.h>

#include<fstream>

#include<stdlib.h>

#include<iomanip>

//record specification

using namespace std;

fstream data,indx;

char pri[125][15]; // to hold primary keys

int ind[125],count=0;

void sort1();

class student

{

public :

char Regno[10], Name[25], Address[50], Sem[5], Branch[10], College[15];

public:

void Insert();

void Delete();

void Search(char \*a); /\* To display individual record \*/

void Display();/\* to display all records in file-- yet to implement\*/

void LoadIndex();

void Writeindx();

}s;

void student::Insert()

{

char Buffer[100],offset[10];

int pos, high = count;

data.open("stu.txt",ios::app);

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

pos=data.tellg(); // gets the address of that posn

cout<<"ENTER RECORD DETAILS\n";

cout<<"RegNo : ";

cin>>s.Regno;

for(int low=0;low<=high;)

{

int mid = (low+high)/2;

int k = atoi(pri[mid]), m = atoi(s.Regno);

if(k == m)

{

cout<<setw(44)<<"DUPLICATE RECORD !!!";

data.close();

return;

}

else if(m < k)

high = mid-1;

else if(m > k)

low = mid + 1;

} // to search for duplicate usn -- can avoid

cout<<"Name : "; cin>>s.Name;

cout<<"Address : "; cin>>s.Address;

cout<<"Sem : "; cin>>s.Sem;

cout<<"Branch : "; cin>>s.Branch;

cout<<"College : "; cin>>s.College;

sprintf(Buffer,"|%s|%s|%s|%s|%s|%s|#",s.Regno,s.Name,s.Address,s.Sem,s.Branch,s.College);

data<<Buffer;

strcpy(pri[count],s.Regno);

ind[count]=pos;

count++;

cout<<count;

data.close();

s.Writeindx();

}

void student::Writeindx()

{

char buffer[100];

int i;

indx.open("inx.txt", ios::out);

sort1();

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

{

//itoa(ind[i],buffer,10);

//sprintf(buffer, "%d", ind[i]);

indx<<"|"<<pri[i]<<"|"<<ind[i]<<"|"<<"#";

}

indx.close();

}

void sort1()

{

char temp[20];

int tempind,k,i,j,l;

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

{

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

{

k = atoi(pri[i]);

l = atoi(pri[j]);

if(k > l)

{

strcpy(temp,pri[i]);

strcpy(pri[i],pri[j]);

strcpy(pri[j],temp);

tempind=ind[j];

ind[j]=ind[i];

ind[i]=tempind;

}

}

}

}

void student::Search(char \*reg)

{

int high = count,flag1=0;

char Buffer[100];

data.open("stu.txt",ios::in);

for(int low=0;low<=high;)

{

int mid = (low+high)/2;

int k = atoi(pri[mid]), m = atoi(reg);

if(k == m) // or else use strcmp()==0

{

flag1 = 1;

data.seekg(ind[mid],ios::beg);

data.getline(Buffer,100,'#');

sscanf(Buffer,"|%[^|]|%[^|]|%[^|]|%[^|]|%[^|]|%[^|]#",s.Regno,s.Name,s.Address,s.Sem,s.Branch,s.College);

cout<<endl<<"RECORD DETAILS"<<endl;

cout<<endl<<"URN : "<<s.Regno;

cout<<endl<<"NAME : "<<s.Name;

cout<<endl<<"ADDRESS : "<<s.Address;

cout<<endl<<"SEMESTER : "<<s.Sem;

cout<<endl<<"BRANCH : "<<s.Branch;

cout<<endl<<"COLLEGE : "<<s.College<<"\n";

data.close();

return; // or use break stmt here

}

else if(m < k)

high = mid-1;

else if(m > k)

low = mid + 1;

}

if(flag1==0)

cout<<endl<<"RECORD DOES NOT EXISTS\n";

data.close();

}

void student::Delete()

{

char reg[20],Buffer[100],usn[15];

int high = count,flag=0;

cout<<endl<<"ENTER USN: ";

cin>>reg;

data.open("stu.txt",ios::in | ios::out);

for(int low=0;low<=high;)

{

int mid = (low+high)/2;

int k = atoi(pri[mid]), m = atoi(reg);

if(k == m)

{

flag=1;

data.seekg(ind[mid],ios::beg);

data.getline(Buffer,100,'#');

Buffer[0]='\*';

pri[mid][0]='\*';

data.seekg(ind[mid],ios::beg);

data<<Buffer;

cout<<endl<<"RECORD DELETED";

data.close();

s.Writeindx();

return;

}

else if(m < k)

high = mid-1;

else if(m > k)

low = mid + 1;

}

if(flag==0)

{

cout<<endl<<"RECORD NOT FOUND !!!";

}

data.close();

}

void student :: Display()

{

char Buffer[100];

data.open("stu.txt",ios::in);

char b[100];

while(!data.eof())

{

data.getline(Buffer,100,'#');

if(data.eof())

break;

if(Buffer[0]!='\*')

{

sscanf(Buffer,"|%[^|]|%[^|]|%[^|]|%[^|]|%[^|]|%[^|]#",s.Regno,s.Name,s.Address,s.Sem,s.Branch,s.College);

cout<<s.Regno<<" "<<s.Name<<" "<<s.Address<<" "<<s.Sem<<" "<<s.Branch<<" "<<s.College<<endl;

}

else

continue;

}

data.close();

}

void student::LoadIndex()

{

char buffer[100],temp[100];

count=0;

indx.open("inx.txt", ios::in);

while(indx)

{

indx.getline(buffer,100,'#');

if(indx.eof())

break;

sscanf(buffer,"|%[^|]|%[^|]",pri[count],temp);

ind[count]=atoi(temp);

count++;

}

indx.close();

}

int main()

{

s.LoadIndex();

fstream f1,f2;

int choice;

char usn[20];

for(;;)

{

cout<<"\n1.Add record \n2.search record \n 3.deleterecord\n4.display record\n";

cin>>choice;

switch(choice)

{

case 1:s.Insert();

break;

case 2:cout<<endl<<"ENTER UsN: ";

cin>>usn;

s.Search(usn);

break;

case 3:s.Delete();

break;

case 4: s.Display();

break;

default: exit(0);

}

}

}

**PROGRAM 6**

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

**Source Code:**

#include<iostream>

#include<stdio.h>

#include<string.h>

#include<fstream>

#include<stdlib.h>

#include<iomanip>

//record specification

using namespace std;

fstream data,indx,secind;

char pri[125][15],sec[125][40],usn[125][20];

int ind[125],count,count1,i,j;

void sort1(), sort2();

int bsearch(int \*, int \*);

class student

{

public :

char Regno[10], Name[25], Address[50], Sem[5], Branch[10], College[15];

public:

void Insert();

void Delete();

void Search(); /\* To display individual record \*/

void LoadIndex();

void Writeindx();

}s;

void student::Insert()

{

char Buffer[100],offset[10];

int pos, high = count;

data.open("stu.txt",ios::app);

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

pos=data.tellg(); // gets the address of that posn

cout<<"ENTER RECORD DETAILS\n";

cout<<"RegNo : ";

cin>>s.Regno;

for(int low=0;low<=high;)

{

int mid = (low+high)/2;

int k = atoi(pri[mid]), m = atoi(s.Regno);

if(k == m)

{

cout<<setw(44)<<"DUPLICATE RECORD !!!";

data.close();

return;

}

else if(m < k)

high = mid-1;

else if(m > k)

low = mid + 1;

} // to search for duplicate usn -- can avoid

cout<<"Name : "; cin>>s.Name;

cout<<"Address : "; cin>>s.Address;

cout<<"Sem : "; cin>>s.Sem;

cout<<"Branch : "; cin>>s.Branch;

cout<<"College : "; cin>>s.College;

sprintf(Buffer,"|%s|%s|%s|%s|%s|%s|#",s.Regno,s.Name,s.Address,s.Sem,s.Branch,s.College);

data<<Buffer;

strcpy(pri[count],s.Regno);

ind[count]=pos;

strcpy(sec[count1],s.Name);//sec record

strcpy(usn[count1],s.Regno);

count++;

count1++;

s.Writeindx();

data.close();

}

void student::Writeindx()

{

char buffer[100];

int i;

indx.open("inx1.txt", ios::out);

secind.open("sec.txt",ios::out);

sort1();

sort2();

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

{

indx<<"|"<<pri[i]<<"|"<<ind[i]<<"|"<<"#";

}

for(i=0;i<count1;i++)

{

secind<<"|"<<sec[i]<<"|"<<usn[i]<<"|"<<"#";//sec record writing

}

indx.close();

secind.close();

}

void sort1()

{

char temp[20];

int tempind;

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

{

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

{

if(strcmp(pri[i],pri[j])>0)

{

strcpy(temp,pri[i]);

strcpy(pri[i],pri[j]);

strcpy(pri[j],temp);

tempind=ind[j];

ind[j]=ind[i];

ind[i]=tempind;

}

}

}

}

void sort2()

{

char temp[40];

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

{

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

{

if(strcmp(sec[j],sec[i])<0)

{

strcpy(temp,sec[i]);

strcpy(sec[i],sec[j]);

strcpy(sec[j],temp);

strcpy(temp,usn[i]);

strcpy(usn[i],usn[j]);

strcpy(usn[j],temp);

}

}

}

}

void student::Search()

{

int flag=0,l,i;

char pos[10];

char Buffer[100],name[20],usn[20],temp[50], ename[20], reg[20], Buf1[50];

cout<<"enter the name to be searched";

cin>>ename;

data.open("stu.txt",ios::in);

secind.open("sec.txt",ios::in);

indx.open("inx1.txt", ios::in);

while(!(secind.eof()))

{

secind.getline(temp,50,'#');

if(secind.eof())

break;

sscanf(temp, "|%[^|]|%[^|]|#", name, usn);

if(strcmp(name,ename)==0)

{

cout<<name<<" "<<usn<<endl;

flag=1;

}

}

if(flag==0)

{

cout<<endl<<"RECORD DOES NOT EXISTS\n";

data.close();

indx.close();

secind.close();

return;

}

cout<<"enter usn to display the record";

cin>>reg;

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

{

indx.getline(Buf1,50,'#');

sscanf(Buf1, "|%[^|]|%[^|]|#", usn, pos);

if(strcmp(usn,reg)==0)

{

l= atoi(pos);

data.seekg(l);

data.getline(Buffer,100,'#');

sscanf(Buffer,"|%[^|]|%[^|]|%[^|]|%[^|]|%[^|]|%[^|]#",s.Regno,s.Name,s.Address,s.Sem,s.Branch,s.College);

cout<<s.Regno<<s.Name<<" "<<s.Address<<" "<<s.Sem<<" "<<s.Branch<<" "<<" "<<s.College<<endl;

break;

}

}

data.close();

indx.close();

secind.close();

return;

}

void student::Delete()

{

char Buffer[100];

int flag=0,mid,mid1;

data.open("stu.txt",ios::in | ios::out);

flag = bsearch(&mid1,&mid);

if(flag==0)

cout<<endl<<"RECORD DOES NOT EXISTS\n";

else

{

data.seekg(ind[mid1],ios::beg);

data.getline(Buffer,100,'#');

Buffer[0]='\*';//delete stu rec

pri[mid1][0]='\*';//delete pri indx

sec[mid][0]='\*';//delete sec indx

data.seekg(ind[mid1],ios::beg);

data<<Buffer;//write deleted record

cout<<endl<<"RECORD DELETED";

s.Writeindx();

}

data.close();

}

int bsearch(int \*mid1,int \*mid)

{ int low=0, high=count, low1=0, high1=count1, k, m;

char name[20];

cout<<"Enter name \n";

cin>>name;

for(low=0;low<=high;)

{

\*mid = (low+high)/2;

if(strcmp(sec[\*mid],name) == 0)

{

for(low1=0;low1<=high1;)

{

\*mid1 = (low1+high1)/2;

k = atoi(pri[\*mid1]), m= atoi(usn[\*mid]);

if(k == m)

{

return 1;

}

else if(m < k)

high1 = \*mid1-1;

else if(m > k)

low1 = \*mid1 + 1;

}

}

else if(strcmp(sec[\*mid],name) > 0)

high = \*mid-1;

else if(strcmp(sec[\*mid],name) < 0)

low = \*mid + 1;

}

return 0;

}

void student::LoadIndex()

{

char buffer[100],temp[100];

count=0;

indx.open("inx1.txt", ios::in);

while(indx)

{

indx.getline(buffer,100,'#');

if(indx.eof())

break;

sscanf(buffer,"|%[^|]|%[^|]",pri[count],temp);

ind[count]=atoi(temp);

count++;

}

indx.close();

}

int main()

{

s.LoadIndex();

int choice;

for(;;)

{

cout<<"\n1.Add record \n2.search record \n 3.deleterecord\n";

cin>>choice;

switch(choice)

{

case 1:s.Insert();

break;

case 2:s.Search();

break;

case 3:s.Delete();

break;

default: exit(0);

}

}

}

**PROGRAM 7**

**Write a 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.**

**Source Code:**

#include<iostream.h>

#include<string.h>

#include<conio.h>

#include<fstream.h>

#include<stdlib.h>

void writelists()

{

fstream out1,out2;

int i,m,n;

char name[20];

out1.open("input1.txt",ios::out);

out2.open("input2.txt",ios::out);

if((!out1)||(!out2))

{

printf("file creation error");

exit(0);

}

cout<<"enter the number of names u want to enterin file 1";

cin>>m;

cout<<"please enter the names in ascending order";

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

{

cin>>name;

out1<<name;

out1<<"\n";

}

cout<<"enter the number of names u want to enterin file 2";

cin>>n;

cout<<"please enter the names in ascending order";

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

{

cin>>name;

out2<<name;

out2<<"\n";

}

out1.close();

out2.close();

}

void main()

{

char list1[100][20],list2[100][20];

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

fstream out1,out2,out3;

writelists();

out1.open("input1.txt",ios::in);

out2.open("input2.txt",ios::in);

out3.open("output1.txt",ios::out);

if((!out3)||(!out1)||(!out2))

{

printf("file creation error");

exit(0);

}

while(!out1.eof())

{

out1.getline(list1[m],20,'\n');

cout<<list1[m]<<" ";

m++;

}

cout<<"\n";

while(!out2.eof())

{

out2.getline(list2[n],20,'\n');

cout<<list2[n]<<" ";

n++;

}

cout<<"\n";

m--;

n--;

cout<<"\n Elements common to both files are:\n";

while(i<m &&j<n)

{

if(strcmp(list1[i],list2[j])==0)

{

out3<<list1[i];

out3<<"\n ";

cout<<list1[i]<<"\n";

i++;

j++;

}

else if(strcmp(list1[i],list2[j])<0)

i++;

else

j++;

}

}

**PROGRAM 8**

**Write a program to read k Lists of names and merge them using k-way merge algorithm with**

**k = 8.**

**Source Code:**

#include<iostream.h>

#include<fstream.h>

#include<string.h>

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

class record

{

public : char name[20],usn[20];

}rec[20];

fstream file[8];

int no;

char fname[8][8]={"1.txt","2.txt","3.txt","4.txt","5.txt","6.txt","7.txt","8.txt"};

void merge\_file(char\* file1,char\* file2,char\* filename)

{

record recd[20];

int i,k=0;

fstream f1,f2;

f1.open(file1,ios::in);

f2.open(file2,ios::in);

while(!f1.eof())

{

f1.getline(recd[k].name,20,'|');

f1.getline(recd[k++].usn,20,'\n');

}

while(!f2.eof())

{

f2.getline(recd[k].name,20,'|');

f2.getline(recd[k++].usn,20,'\n');

}

int t,y;

record temp ;

for(t=0;t<k-2;t++)

for(y=0;y<k-t-2;y++)

// cout<<recd[y].name<<recd[y+1].name;

if(strcmp(recd[y].name,recd[y+1].name)>0)

{

temp=recd[y];

recd[y]=recd[y+1];

recd[y+1]=temp;

}

// cout<<recd[y].name<<recd[y+1].name;

fstream temp1;

temp1.open(filename,ios::out);

for(t=1;t<k-1;t++)

temp1<<recd[t].name<<"|"<<recd[t].usn<<"\n";

f1.close();

f2.close();

temp1.close();

return;

}

void kwaymerge()

{

int i,k=0;

char filename[7][20]={"11.txt","22.txt","33.txt","44.txt","111.txt","222.txt","1111.txt"};

for(i=0;i<8;i+=2)

{

merge\_file(fname[i],fname[i+1],filename[k++]);

}

k=4;

for(i=0;i<4;i+=2)

{

merge\_file(filename[i],filename[i+1],filename[k++]);

}

merge\_file(filename[4],filename[5],filename[6]);

return;

}

int main()

{

int i,j,no;

cout<<"enter the no.of records\n";

cin>>no;

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

{

cout<<"\n enter the deatils of file:"<<i<<endl;

file[i].open(fname[i],ios::out);

for(j=0;j<no;j++)

{

cout<<"NAME:";

cin>>rec[j].name;

cout<<"USN:";

cin>>rec[j].usn;

file[i]<<rec[j].name<<"|"<<rec[j].usn<<"\n";

}

}

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

file[i].close();

kwaymerge();

fstream result;

result.open("1111.txt",ios::in);

cout<<" sorted records are\n";

char name[20],usn[20];

for(i=0;i<(no\*8);i++)

{

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

result.getline(usn,20,'\n');

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

}

return 0;

}