DESIGN AND DEVELOPMENT OF A SERIAL DATABASE : A CASE STUDY

Ms. Lakshmi R & Prof. Ravichandra Rao I. K.

ABSTRACT

Development of a library database is a prerequisite for the automated library system. The commercial softwares available are neither inadequate to any particular library for specific functions nor economical. Also, most of these packages hardly adopted international standards. To overcome these problems, an attempt has been made to develop a software for create and maintennance of serial database. This paper discusses the importance of serial databases along with the usability of the programs developed in C++. Programs adopts CCF and ISO 2709.

1. Introduction

Computerzation of libraries has reached to such an extent that libraries are now depend mainly on the international databases, even for their day-to-day services. Especially, the 'Internet fever' has led us to believe that digital libraries are only the solutions to our problem. In this circumstances we in India tend to give least importance to computerization of catalogues of books and serials. However simple it may look, it is difficult to develop such machine-readdable catalogues of books and serials (or library databases) due to certain local constraints.

In this paper an attempt has made to explain the importance of local library databases, wirh emphasis on serials. How to go about developing a serial database ? A program in C++is being developed at DRTC in this regard. Its features along with the limitations of the program are discussed.

1.1 What is a Databases?

A databases is a collection of inter-related data, about an item stored torether with controlled redundancy to serve one or more applications; the data are stored in such a way that they are independent of programs as well as hardwar. A common and controlled approach is used in adding new data, in modifying and retrieving existing data. The information stored in a database can be used in different ways for different purposes; it implies that tailor-made information may be obtained from a database.

1.2 Why Database for Libraries ?

In this juncture, a basic question arises why have to develop a database in the context of a library? The following logical reasons will support strongly the necessity of a database in a library.

a) A database or a machine-readable record may be used repetitively for several purpose which in-turn saves effort, time and resources considerably compared to those involved in manual process.

b) It helps in avoiding duplication of work.

c) It helps us to have an effective control over the entire collection which further helps in collection development.

d) It helps to improve the existing services (regularity, accuracy of a service) as well as to introduce new services.

e) It can be accessed from the work station in a LAN environment.

f) It promotes effective resoures sharing among the libraries through a network which leads to access to information in a cost effective way.

Several databases (database of books, vendores, circulation transactions, binding details) may have to be developed in a library. In this paper, only creation and development of a serial database is given importance.

2. Serial database

2.1 What is Serial Database ?

A database is a collection of interrelated data regarding serials to serve many applications. The different data elements which constitute a record may be journal title, frequency, ISSN, publisher name & address, publication date, beginning yeat

of subscription, library holdings, binding and other information,

Some of the data elements or many more may be considered depending upon the purpose for which database is developed.Generally, one may have to follow certain guidelines to include the data elements in a database. For seloection of data elements and their respective tags, CCF (1) guides us in the right direction. The structure of the record may be in conformity with ISO 2709.

2.2 An Ideal Serial Database

Development of a serial database is pre-requisite to develop an automated serial control system. An ideal serial database in the context of automated serial control system has to handle serial information and maintain holding list. To achieve these objectives, the systems must perform certain mandatory functions. The check-list of functions of an ideal automated serial control is as follows (3).

2.2.1 Ordering and subscription control

*pre-order searching to establish its availability
*creation of orders
*system-generated data of order, order number etc.,
*provision of handling supplier data
*provision for entering frequency, volume, issue information to enable system to predict forthcoming issuse
*produce printed orders as and when required with librarydefined text and free messages for suppliers
*immediate updating of fund information
*to handle reports from suppliers
*transfer order directly to subscription agent, if necessary

2.2.2 Check-in

*retrieve record on a variety of keys
*display of issue expected; check-in by single keystroke if correct
*check-in of multiple copies on a single screen
*provision for partial receipts
*provision for viewing other predicted issues if issue in hand is not the issue expected; provision for entering issue not predicted,e.g. index or supplement,etc,.
*provision for marking items for claiming of damaged issues

2.2.3 Routing

*provision of routing lists individually for titles checked-in,
*creation and maintenance of routing lists for specfic copies of serials using user file
*online access to lists by serial title, for list of recipients, or by recipients, for list of titles routed
*priority levels for individuals on routing lists

2.2.4 Claiming

*identification of missing or overdue issues based on predicted expected issues,, together with library-defined claim period (by title or supplier)
*notification of overdue or missing issues to library for review and authorization of claim or automating claim

2.2.5 Binding

*indication of when a title is ready for binding,e.g.specified issue/volume number
*prepare list items ready binding with all details - - instructions, colour, lettering, type of binding etc,.

2.2.6 Fund accounting

*provision of fund/cost centres for allocation of total fund, and committed and actual expenditure
*provision for currency conversion
*invoice processing and payment authorization
*warning if fund not available and immediate updating of fund information

2.2.7 Enquiries

*sccess on a variety of keys to all levels of serials information; holdings; issue expected, received, missing/overdue, and claimed
*latest issues received displayed on OPAC (Online Open Access Catalouge)

2.2.8 Report and Statistics

*standard reports relating to missing/overdue issues and claims; subscriptions due for renewal; supplier performance; bindery performance; and fund reports

3 Computer program

An attempt has been made to develop a software at DRTC to perform above functions for automated serial control system. This project has been launched as the existing commercial softwares do not serve our purpose specifically. This is also the experience of many institutions. (2) and also CDS/ISIS is adequate only for information retrieval purpose and not for library automation. Since CDS/ISIS cannot handle both field indicator and sub-field indicator simultaneously, as well as it cannot handle multiple detabases at a time, another software has to be developed from the scratch. Till time, only part of the project has been completed and its features are discussed below.

3.1 Important Features of the program :

As a part main program of four different modules have been completed. They are

* Creation of Master File
* Upgradation of Master File
* Indexing Services

* Printing Services

3.1.1 Creation of Master File

3.1.1.1 Purpose:
The main purpose of the program is to, create a multi-purpose file consisting of bibliographic and other information about serials; the record format is in conformity with ISO 2709. This file can be used as input for

A. Ordering and subscription control
1. Pre-order searching to check for a title.
2. Creation of orders/subscription letters.

B. Information Retrieval
1. To retrieve bibliographic information of a required serial, including details of the publisher/distrubutor.
2. Searching the database through different keys viz
serial title, publisher, agent etc,.

C. Printing
1. To print the holding file of serials in different formats.
2. To print the list of journals subscribed, by country, by language, by subject etc,.
3. To print missing issues along with journal titles.
Basically the program called create.cpp is written to accept data elements one-by-one, record status and bibliographic level form the keyboard. After the construction of leader and directory, data elements along with the leader and directory are written to a file. A record thus consists of leader, directory, data and record separator.

3.1.1.2 Sample Input
The Syntax of input is
<tag><field indicator><sub-field indicator><data><sub-field indicator><data><...> and press enter.
For example,

001 INTELI-1982
020 INDIAN STATISTICAL INSTITUTE BANGALORE CENTRE
040 ^aeng
201 ^aInformation Technology and Libraries
1010730-9295
230 Infor Tech Lib
400 ^aLibrary and Information Technology Association
^bDivision of ALA^c50E Huron Street
401 ^aUSA^bChicago, IL
402 ^a60611
440 ^a1968^b1982
520 Quarterly
450 ^aV.1.1982+^cV.4.1985

3.1.1.3 Sample Output
Out put is stored in a file named *.mst,where * stands for name of the database. A Sample output is

00453n0s0022001680004520001001200000020004600012040000060005820100390006410100100010323000150011340000840012840100190021240200080023144000130023952000100025245000220026 2INTELI-1982#INDIAN STATISTICAL INSTITUTE BANGALORE CENTRE#^aeng#^aInformation Technology and Libraries#0730-9295#Infor Tech Lib#^aLibrary and Information Technology Association^bDivision of ALA^c50E Huron Street#^aUSA^bChicago,IL#^a60611#^a1968^b1982#quarterly#^aV.1.1982+^cV.4.1985##

Using this output file as an input, we can generate Several services. For example, indexing and printing Services. These are discussed below.

Index for a master file can be generated using the program called index.cpp by keying-in the corresponding tag of the desired field to be indexed from the key board.

Using print.cpp program, the records can be printed with different options. The main option is selecting the number of records i.e., selecting all the records or printing selected records. The sub-options refers to sending to the records to different output devices viz., console, printer and to a file.
Eg, a print format of a record is

Record id : INTELI-1982
Source : INDIAN STATISTICAL INSTITUTE BANGALORE CENTRE
Language : eng
Journal Title : Information Technology and Libraries
ISSN : 0730-9295
Spine Title : Infor Tech Lib
Publisher Details : Library and Information Technology Association^bDivision of ALA^c50E Huron Street
Country : USA^bChicago, IL
Pin : 60611
Publication Date : 1968^b1982
Frequency : Quarterly
Holding : V.1.1982+^cV.4.1985

In the upgradation of the master file module, three options are made available. They are adding a new record, deleting and modifying the exixting record. At present programming for adding and deleting a record has been completed. The programs can be easily modified to accomodate more than twenty-five fields without much efforts; otherwise as it is, it accepts any where between 1 to 25 data elements per record.

The features such as modifying a record, fund accounting and check-in of journals is expected to be completed in the next six to nine months. The source code are given in Appendix I.

4. CONCLUDING REMARKS

With an objective to develop a software for serial control, attempts were made to write a few programs in C++ for creation and operation of serial database. Programs were tested with 150 records of the serials which are subscribed at ISIBC library. Also, inclusion of field indicators and elimination of both field and sub-field indicators in the printed output has to be incorporated.

5. REFERENCES

1. Simons (Peter) and Hopkinson (Alan), ed. CCF : the Common Communication Format, 2nd ed. GIP & UNISIST, UNESCO, Paris, 1988.

2. Patel (D K) and Joshipura (Smita D). Computer-based periodicals management system in SAC library. (DESIDOC Bulletin of Information Technology), 15(3), 1995)

3. Ravichandra Rao (I K). Library Automation. Wiley Eastern. New Delhi, 1990

4. Serials Control. (Library Technology Report, Jan/ Feb, Section 3, 1992.)

APPENDIX I

```
/* This program(create.cpp) converts the input data element into ISO 2709 format and adds that record to
the existing or new file */
include <fstream.h>
#include <conio.h>
#include <string.h>

void main()
{
struct input_type
                {
                char data[150];
                }
                data[25];
                /* THIS IS FOR STORING INPUT DATA */
Struct label_type
                {
                int rec_len;
                char rec_status;
                int spacel;
                char bib_level;
                int space2;
                int indicator_len;
                int sub field identifier_len;
                int base_address;
                int space3;
                int len_ datafield;
                int len_start_char_position;
                int len_of_identifier;
                int space4;
                }
                lable;
                /* THIS IS FOR STORING LABLE INFORMATION */
struct dir_type
        {
        char tag[4];
        int len_datafield;
        int start_char_position;
        }
        dir[25];
                /* THIS IS FOR STORING DIRECTORY INFORMATION */
        ofstream out("text.txt",ios: :app);
        //OPENING THE OUTPUT FILE TEXT.TXT FOR WRITING PURPOSE

        //VARIABLE DECLARATION
        char ch='y',ch1='y',data1[1000];
        int i=0,m=0,j,k,n,s=0,len,len1[25];
        int count=1,count1=1;
        char end[]={'e','n','d','\0'};

        //INITIALIZING PART OF THE LABEL

        lable.spacel=0;
        lable.space2 = 0;
        label.indicator_len =2;
        label.sub_field_identifier_len =2;
        lable.space3 = 0;

        //INITIALIZING PART OF THE DIRECTORY
```

```
        for (i=0;i<25;i++)
        {
        dir[i].len_datafield=0;
        dir[i].start_char_position=0;
        }

        clrscr();
        while (chl=='y' && m<count1)
                {
                i=0;
                while(1)
                {
                clrscr();
                gotoxy (25,2);
                cout<<"DATA ENTRY SCREEN";
                window(10,10,50,50);
                textcolor(BLACK);
                textcolor(WHITE);
                gotoxy(30,5);
                cout<<"\nEnter data element : ";
                cin.getline(data[i].data,150);//TO PUT DATA ELEMENT
// OF 150 CHARACTERS TO data[i].data
                if   (strcmp(data[i].data,end)==0)  //COMPARING TWO
STRINGS
        goto label;
        else
                {
                i++;
                count++;
                countinue;
                }
        {
                label : gotoxy(12,13);
                cout<<"\nEnter the record status New/Deleted/:";
                cin>>label.rec_status;
        gotoxy(1,16);
        cout<<"Enter the bibliographic leval of the record :";
                cin>>label.bib_level;
                k=0;
                for (i=0;i<count-1;i++)
                {
                strncpy (dir[i].tag,data[i].data,3);
        //TO EXTRACT TAG FROM THE DATA ELEMENT
                dir[i].tag[3]='\0';
                len=strlen(data[i].tag,data);
        //TO CALCULATE LENGTH OF DATA ELEMENT
                data[i].data[len]='#';
                data[i].data[len+1]='\0';
len[i]=len-2;
dir[i].len_datafiled=len1[i];//TO CALCULATE LENGTH OF THE DATA
FIELD
dir[i+1],start_char_position=dir[i].start_char_position      +
dir[i].len_datafield;

//TO CALCULATE STARTING CHARACTER POSITION
        do
         {
           for(j=3;j<len;j++)
                {
                data1[k] =data[i].data[j];//TO PUT ALL DATA ELEMENT INTO
```

```
}
cout<<"\nDO YOU WANT TO ENTER another record? ";
++m;
++count1;
ch1=getche();
}

else {cout<<"unable to open the file xyz.txt";}
}
out.close(); //CLOSING THE O/P FILE
}//main
/* This program index.cpp indexes the master file for required
tag keyed-in from the key bord */

#include <fastream.h>
#include <stdlib.h>
#include <conio.h>

int base, tag, data_len, start_char,k, count;
char ch[700], ch1[13];

ofstream fout("tit.ndx");

void main()
{
int index;
char dat[150];

ifstream fin("text.txt");

int base_address();
int extract_tag();
int extract_data_len();
int extract_start_char_position();

void extract_data():

clrscr();
cout<<"Enter the tag # for which index has to be created : ";
cin>>index;

k=1;

fin.getline(ch, 700,'\n');

do
{
base=base_address();
int i=24;
count=i;
again : tag=extract_tag();
if(tag==index)
{
data_len=extract_data_len();
start_char=extract_start_char_position();
extract_data();
k++;
fin.getline(ch/ 700,'\n');
continue;
}
```

```
else count+=12
goto again;
}while(!fin.eof());
fin.close();
fout.close();
}

int base_address();
{
char base_add[5];
int j=0;
for(int i=13; i<17; i++)
        {
         base_add[j]=ch[i];
         j++;
        }
        base_add[j]='\0';
        return(atoi(base_add));
}
int extract_tag()

{
char tag1[4];
int j=0;

for(int i=count; i<count+3; i++)
{
tag1[j]=ch[i];
j++;
}
tag1[j]='\0';
return (atoi(tag1));
}
int extract_data_len()
{
char data_len[5];
int j=0;
for(int i=count+3; i<count+7; i++)
{
        data_len[j]=ch[i];
        j++;
}
        data_len[j]='\0';
        return (atoi(data_len));
}
int extract_start_char_position()
{
char start_char[6];
int j=0;
for(int i=count+7; i<count=12; i++)
{
start_char[j]=ch[i];
j++;
}
start_char[j]='\0';
return (atoi(start_char));
}
void extract_data()
{
char data[126], data[125];
```

```
int j=0;
for(int i=base+star_char; i<base+start_char+data_len-1; i++)
{
if(ch[i]=='^')
{
i+=2;
data[j]=' ';
            j++;
            data[j]=ch[i];
            j++;
}
          else
          {
          data[j]=ch[i];
          j++;
          }
}
data[j]='\0';
if(data[0]==' ')
{
int len=j;
j=0;

for(i=1;i<len;i++)
{
          data[j]=data[i];
          j++;
}
data[j]='\0';
fout<<data1<<' '<<k<<end1;
}
else
fout<<data<<' '<<k<<end1;
}
/* Print .cpp constructs the print format and sends the
same to different output devices according to the choice */
#include <fstream.h>
#include <string.h>
#include <conio.h>

int base;
char ch[700],ch1[13],file_name[10];
int dir[20] [3],m,mcount;

struct
          {  char rec_id[20];
char source[50];
            char lang[6];
            char serial_tit[75];
            char issn[11];
            char coden[10];
            char spine_tit[30];
            char pub[125];
            char pub_country[75];
            char pub_pin[30];
char dist[100];
            char dist_country[30];
            char dist_pin[20];
            char pub_date[40];
char freq[20];
```

```
            char holding[125];
          }
          print;
void main(int X, int Y)
{
          int base_address();
          void dir_table();
          void extract_data();
          void print_all_printer();
          viod print_record();
          viod print_all_file();

          char dat[125];
          int i,k=0;

          ifstream fin("text.dat");
          while(!fin)
          {
          cout<<"Cannot open the file text1.dat";
          getch();
          main_scr();
          }
          clrscr();
if(x==1 && y==1) //condition for all records to printer
{
do
{
          fin.getline(ch,700,'\n');
          base=base_address();
          dir_table();
          extract_data();
          print_all_printer();
}while(!fin.eof());
}
if(x==1 && y==2) // condition for all records to console
{
do
{
fin.getline(ch,700,'\n');
base=base_address();
dir_table();
extract_data();
print_record();
getch();
}while(!fin.eof());
}
if(x==1 && y==3) // condition for all records to a file
{
          cout<<"Enter File Name : ";
          cin>>file_name;
          int k=1;
do
{
fin.getline(ch,700,'\n');
base=base_address();
dir_table();
extract_data();
print_all_file();
gotoxy(2,10);
cout<<"Processing.... : "<<k;
```

```
k++;
}while(!fin.eof());
}
fin.close();
}

int base_address() // function to find base address of a record
{
int i,k=0;
char base_add[5];
        for(i=13;i<17;i++)
        {
base_add[k]=ch[i];
k++;
}
base_add[k]='\0';
return(atoi(base_add));
}

void dir_table() // function to construct directory table for
// a record in the matrix form
{
int i=24,j=0;
int count=i;

int matrix0();
int matrix1();
int matrix2();
int m=0;

do
{
for(i=count;i,i<count+12;i++)
{
ch1[j]=ch[i];
j++;
}
ch1[j]='\0';
dir[m] [0]=matrix0();
dir[m] [1]=matrix1();
dir[m] [2]=matrix2();
count+=12;
j=0;
m++;
mcount=m;
}
while(count<=base-12);
}
int matrix0()
{
char tag[4];
int i,j=0;
        for(i=0;i<3;i++)
         {
         tag[j]=ch1[i];
         j++;
         tag[j]='\0';
         j=0;
         return(atoi(tag));
```

```
int matrix1()
{
char data_len[5];
int i,j=0;

        for(i=3;i<7;i++)
{
data_len[j]=ch1[i];
j++;
}
data_len[j]='\0';
j=0;
mks.



        return(atoi(data_len));
}
int matrix2()
{
char start_char_position[6];
int i,j=0;

        for(i=7;i,12;i++)
        {
        start_char_position[j]=ch1[i];
        j++;
        }
start_char_position[j]='\0';
j=0;
return(atoi(start_char_position));
}

void extract_data()

/* function to extract data for all data elements by taking information about tag, starting character and length
of the data elements from the matrix constructed.*/
        {
        char dat[126];
        int tag1,data_len1,start_char1,i,j;

        m=0;
        while(m,mcount)
         {
          int k=0;
          tag1=dir[m] [0];
          data_len1=dir[m] [1];
          start_char1=dir[m] [2];
        if (ch[base+start_char1]=='^')


{
f       o       r

(i=base+start_char1+2;i<base+start_char1+data_len1-1;i++)
                {
                  dat[k]=ch[i];
                  k++;
                }
                dat[k]='\0';
```

```
        }
        else
        {
         for(i=base+start_char1;i<base+start_char1+data_len1-1;i++)
         {
          dat[k]=ch[i];
          k++;
         }
         dat[k]='\0';
}
switch(tag1)
{
case 1 :
        strcpy(print.rec_id,dat);
        break;
case 20 :
        strcpy(print.source,dat);
        break;
case 40 :
        strcpy(print.lang,dat);
        break;
case 201 :
        strcpy(print.serial_tit,dat);
        break;
case 101 :
        strcpy(print.issn,dat);
        break;
case 102 :
        strcpy(print.coden,dat);
        break;
case 230 :
        strcpy(print.spine_tit,dat);
        break;
case 400 :
        strcpy(print.pub,dat);
        break;
case 402 :
        strcpy(print.pub_pin,dat);
        break;
case 420 :
        strcpy(print.dist,dat,dat);
        break;
case 421 :
        strcpy(print.dist_country,dat);
        break;
case 422 :
        strcpy(print.dist_pin,dat);
        break;
case 440 :
        strcpy(print.pub_date,dat);
        break;
case 520 :
        strcpy(print.freq,dat);
        break;
case 450 :
        strcpy(print.holding,dat);
        break;
}
        m++;
```

```
        }
}
void print_record()//function to print the record to the screen
        {       int j=0;
                cout<<" Record_id :"<< print.rec_id<<end1;
                cout<<" Source :"<< print.Source<<end1;
                cout<<" Language : "<<print.lang<<end1
                cout<<" Journal Title : "<<print.serial_tit<<end1;
                while(!print.coden[j]==' ')
                {
                cout<<"                 Coden : "<<print.coden<<end1;
                }
                cout<<"                 Spine Title :"<<print.spine_tit<<end1;
                cout<<"                 Publisher Details : "<<print.pub<<end1;
                cout<<"                  Country : "<<print.pub_country<<end1;
                cout<<"                  Pin : "<<print.pub_pin<<end1;
                while(!print.dist[j]==' ')
                cout<<"                  Distributor Details :  "
                <<print.dist<<end1;
                }
                while(!print.dist_country[j]==' ')
                {
                cout<<"                   Country : "<<print.dist_country<<end1;
                }
                while(!print.dist_pin[j]==' ')
                {
                cout<<"                   Pin :  "<<print.dist_pin<<end1;
                }
                cout<<"                   Publication Date :   "
                <<print.pub_date<<end1;
                cout<<"                   Frequency :  "<<print.freq<<end1;
                cout<<"                   Holding :      "<<print.holding<<end1;
                cout<<end1<<end1;
                }

void print_all_file() // function to write the records to a file
        {
                int j=0;

                ofstream out(file_name, ios::app);

                out<<"  Record_id : "<< print.rec_id<<end1;
                out<<"  Source : "<< print.Source<<end1;
                out<<"  Language : "<< print.lang<<end1;
                out<<"  Journal Title : "<<print.serial_tit<<end1;
                out<<"  ISSN : "print.issn<<end1;
                while(!print.coden[j]==' ')
                {
                out<<"                  Coden : "<<print.coden<<end1;
                }
                out<<"                  Spine Title : "<<print.spine_tit<<end1;
                out<<"                  Publisher Details : "<<print.pub<<end1;
                out<<"                   Country : "<<print.pub_country<<end1;
                out<<"                   Pin : "<<print.pub_pin<<end1;
                while(!print.dist[j]==' ')
                {
                out<<"                   Distributor Details :  "
                <<print.dist<<end1;
                }
                while(!print.dist_country[j]==' ')
```

```
                {
                out<<"   Country : "<<print.dist_country<<end1;

                }
                while(!print.dist_pin[j]==' ')
                {
                out<<"          Pin :  "<<print.dist_pin<<end1;
                }
                out<<"          Publication Date :  "
                <<print.pub_date<<end1;
                out<<"          Frequency :     "<<print.freq<<end1;
                out<<"          Holding :               "<<print,holding<<end1;
                }
                out<<end1<<end1;
out.close();
        }

void print_all_printer() // function to send the records to a
printer
        {
                int j=0;
                ofstream out;
                out.open("PRN");

                out<<"  Record id : "<<print.rec_id<<end1;
                out<<"  Source : "<<print.source<<end1;
                out<<"  Language : "<<print.lang<<end1;
                out<<"  Journal Title : "<<print.serial_tit<<end1;
                out<<"  ISSN : "<<print.issn<<end1;
                while(!print.coden[j]==' ')
                {
                out<<"  Coden : "<<print.coden<<end1;
                }
                out<<"  Spine Title : "<<print.spine_tit<<end1;
                out<<"  Publisher Details : "<<print.pub<<end1;
                out<<"   Country : "<<print.pun_country<<end1;
                out<<"   Pin : "<<print.pub_pin<<end1;
                while(!print.dist[j]==' ')
                {
                out<<"  Distributor Details : "
                <<print.dist<<end1;
                }
                while(!print.dist_country[j]==' ')
                {
                out<<"  Country : "<<print.dist_country<<end1;
                }
                while(!print.dist_pin[j]==' ')
                {
                out<<"  Pin : "<<print.dist_pin<<end1;
                }
                out<<"  Publication Date :  "
                <<print.pub_date<<end1;
                out<<"  Frequency :     "<<print.freq<<end1;
                out<<"  Holding :               "<<print.holding<<end1;
                out<<end1<<end1;
out.close();
        }
```