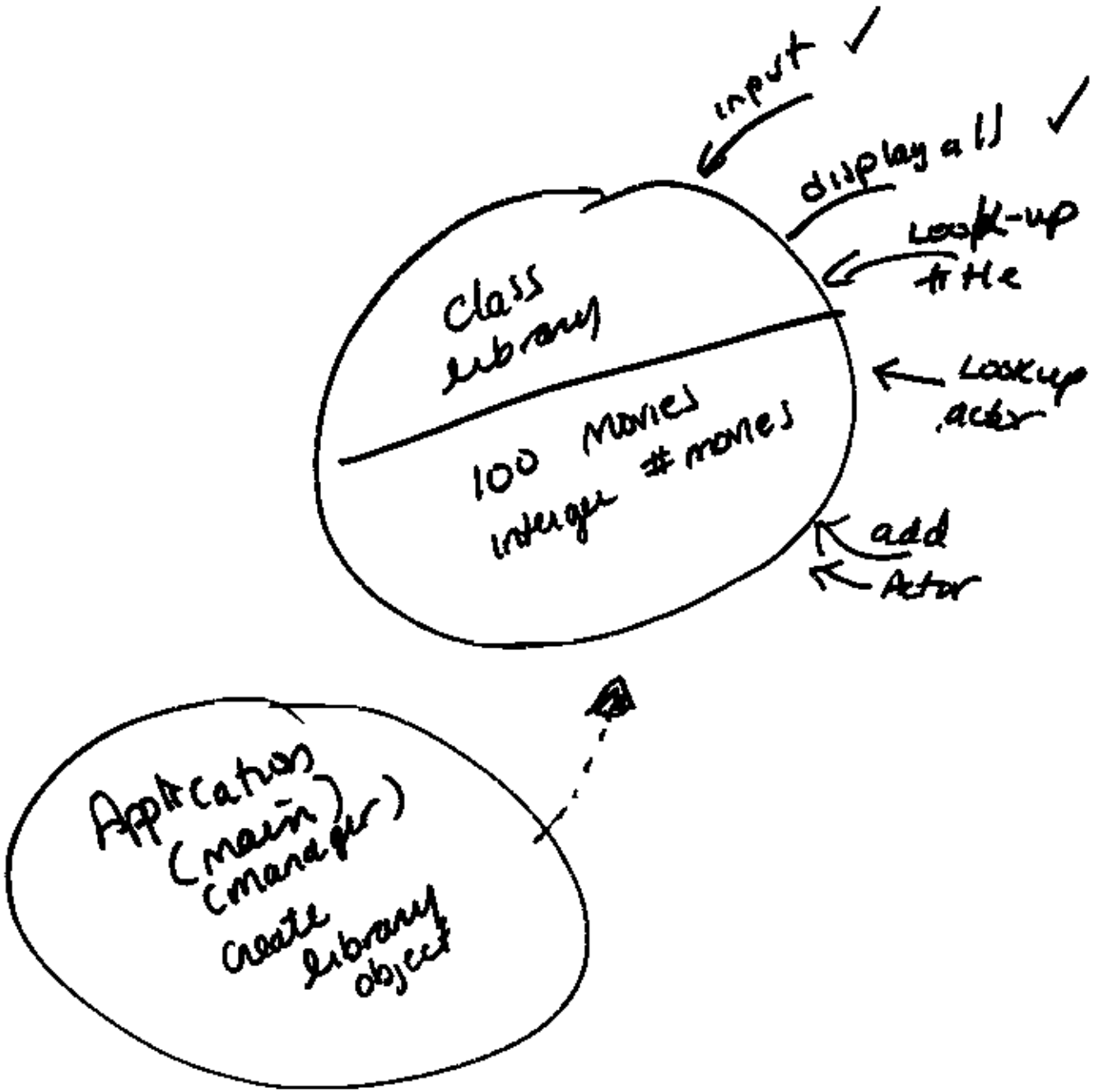

input ✓
display all ✓
Look-up #He
Class Library
100 Movies
integer #movies
Lookup actor
add Actor
Application (main) (manager)
create library object

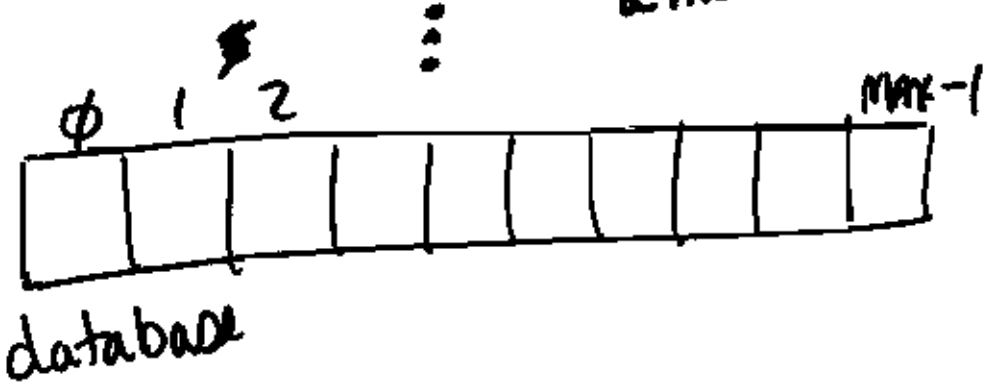
library.cpp

.cpp file

```
int library::add(movie &toadd)
{   if(num_movies >= MAX)
        return 0;
    int i = num_movies;
    strcpy(database[i].title, toadd.title);
    strcpy(database[i].review, toadd.review);
    database[i].num_stars = toadd.num_stars;
    for(int j=0; j < toadd.num_stars; ++j)
    {   strcpy(database[i].stars[j].first,
               toadd.stars[j].first);
        ⋮
```

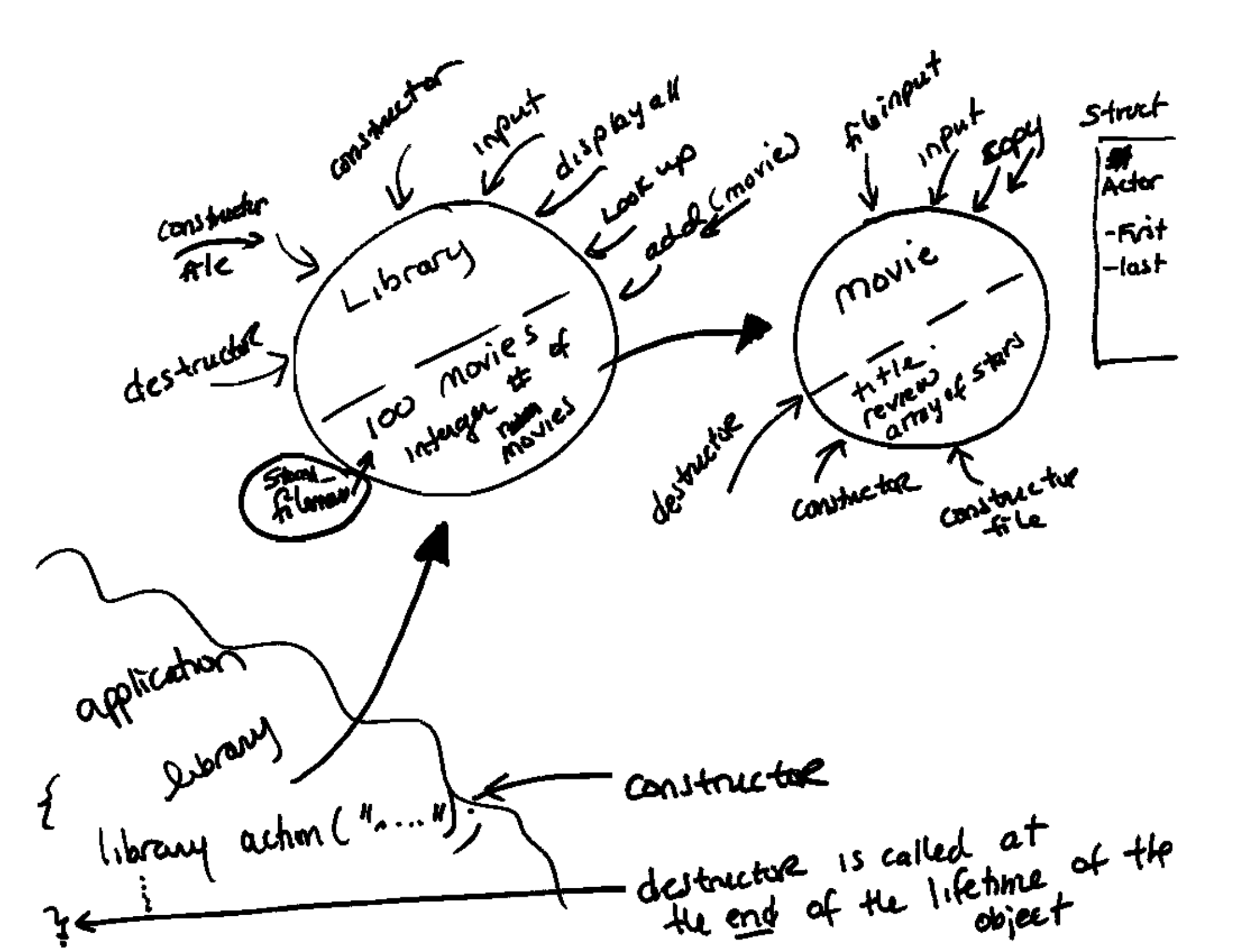
database[i].title → a movie

database[i].stars[j] → actor


store into database[num_movies]

```
            strcpy(database[i].stars[j].~~first~~last
                                        toadd.stars[j].last);
        }
        ++num_movies;
        return num_movies;
    }
```


constructor
input
display all
Look up
add(movie)
constructor
file
destructor
Library
100 movies
integer # of movies
Store_filename
fstream input
copy
Struct
Actor
-First
-last
Movie
title
review
array of stars
destructor
constructor
constructor file
application
library
{
library action("...");
}
constructor
destructor is called at the end of the lifetime of the object

```
int library::input()
{
        if (num_movies >= MAX)
                return 0;
        database[num_movies].input();
                a movie
        return ++num_movies;
}

int movie::input()
{
        cout << "please enter the title ";
        cin.get(title, 131); cin.ignore(100, '\n');
```

```
int movie :: get( char prompt[], char field[], int size)
{
    cout << prompt;
    cin.get( field, size);
    cin.ignore (100, '\n');
    //Error ..... checking
}
```

```
int movie :: input()
{
    get ("please enter title", title, 131);
    get(" " " review", review, 501);
    ~~get(" " " " ".~~
    cout << " How many actors ... ";
    cin >> num-stars; cin.ignore();
    for(int i= 0; i < num-stars; ++ i)
    {   get ("please stars-first", stars[i].first, 21);
        get (" ..... -last ", stars[i].last, 41);
    }
```

```
int library::add(movie &toadd)
{
    if (num_movies >= MAX)
        return 0;
    database[num_movies].copy(toadd);
    return ++num_movies;
}
```

a movie

a movie

```
int movie::copy(movie &toadd)
{
    strcpy(title, toadd.title);
    strcpy(review, toadd.review);
    num_stars = toadd.num_stars;
    for(int i = 0; i < num_stars; ++i)
    {   strcpy(stars[i].first, toadd.stars[i].
                                    first);
        strcpy(stars[i].last,
                    toadd.stars[i].last);

    }
    :
}
```

library.h

```
#include ....
const
struct
class interfaces
(prototypes)
```

```
class store_inventory
{ public:

  private:

};
```

lbrary.cpp

```
#include "inv.h"
store_inventory:: store_inventory()
{  :
}
```

CODE FOR ALL Member functions

main.cpp

```
#include "inv.h"
int main()
{  :
}
```

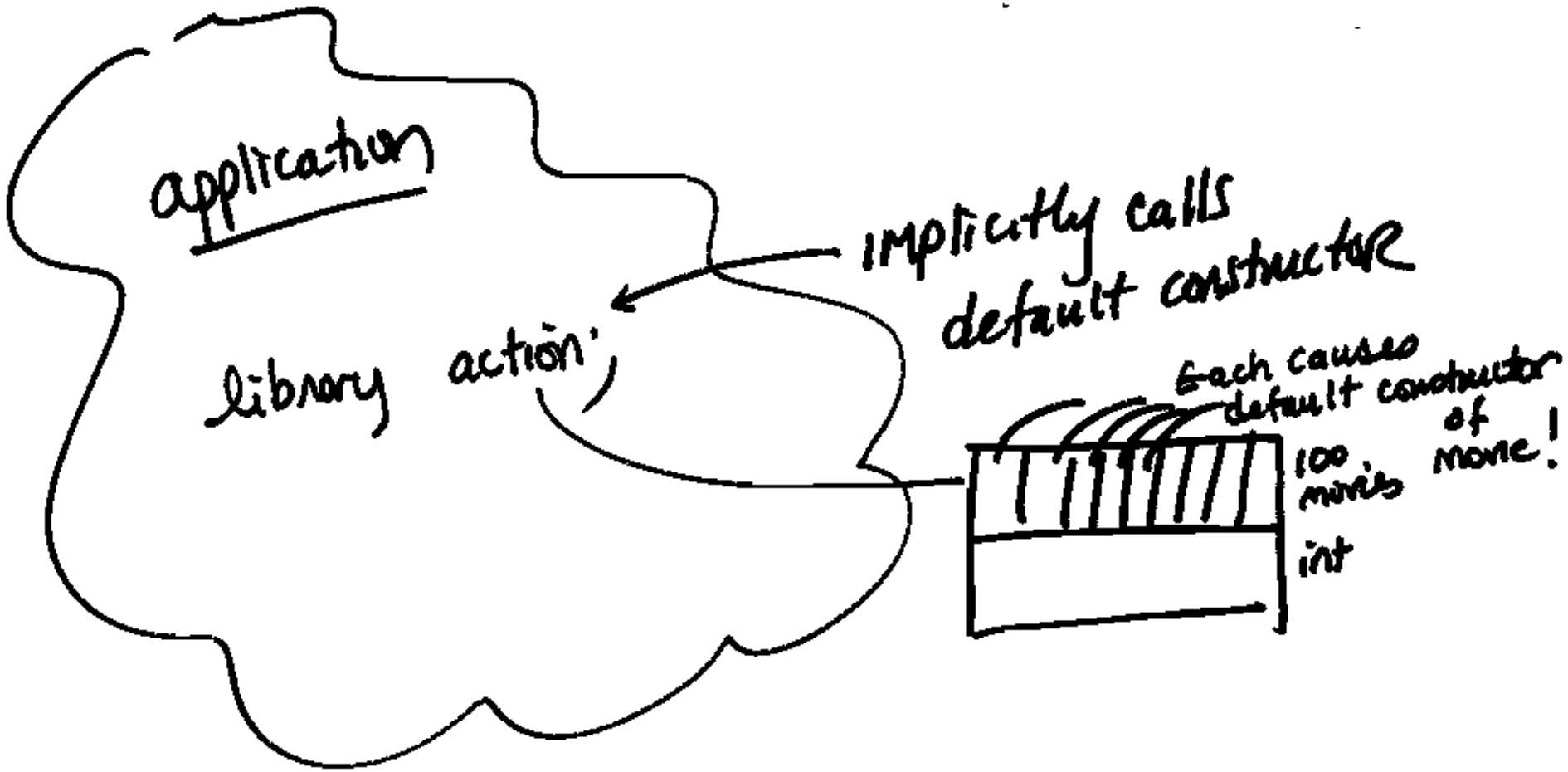
library.h

any other functions

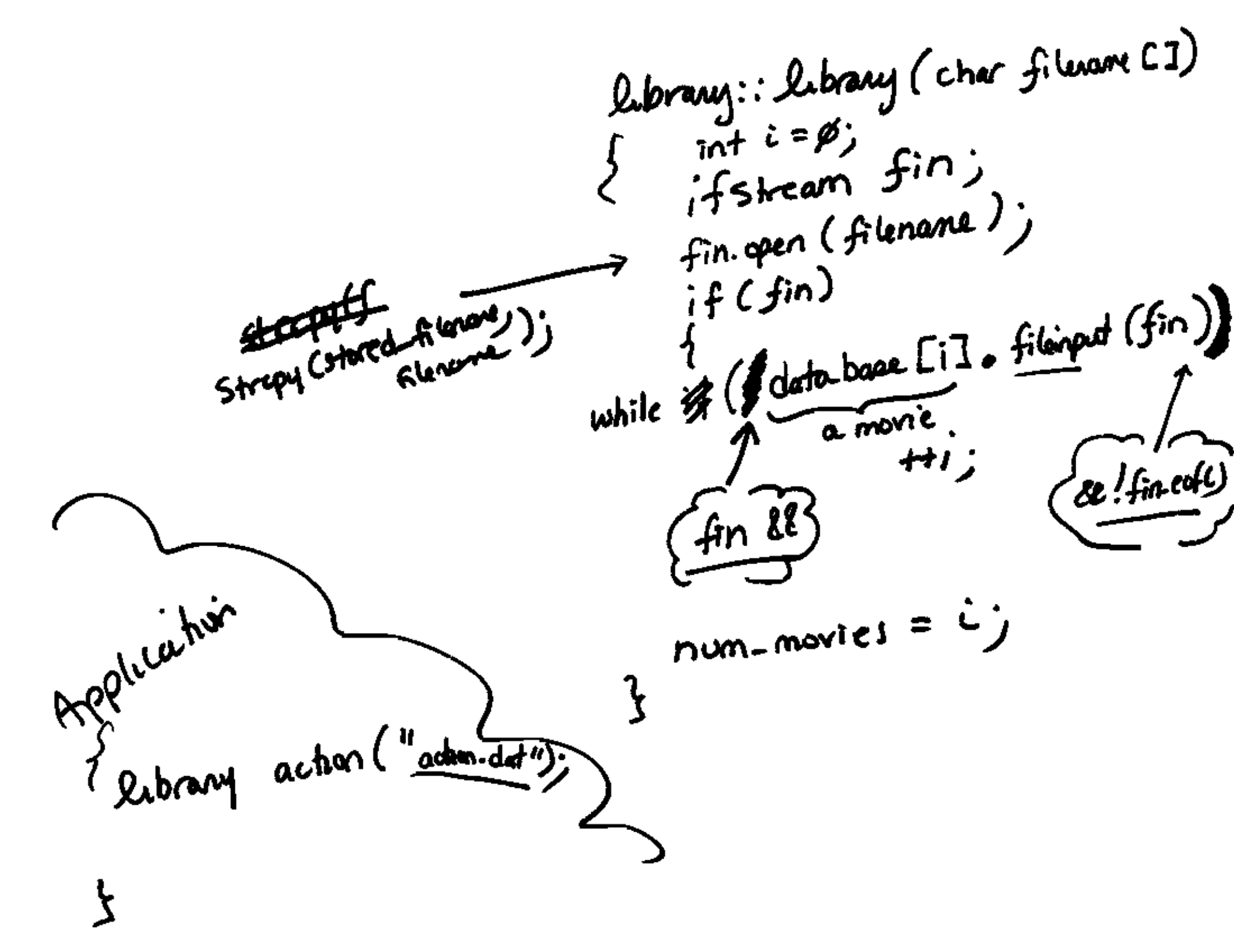
g++ *.cpp

g++ main.cpp inv.cpp

```
Library:: Library()   // default ← RULE!
{
    num_movies = 0;
}
```

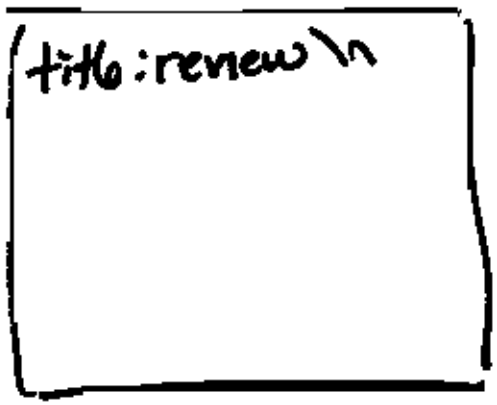


```
movie :: movie()     // default
{
        title[0] = '\0';
        review[0] = '\0';
        num_stars = 0;
}
```


library::library(char filename[])
{
int i=0;
ifstream fin;
fin.open(filename);
if(fin)
{
while((data_base[i].fileinput(fin))
a movie
++i;
fin &&
&& !fin.eof()
num_movies = i;
}
Strcpy(stored_filename, filename);
Application
{
library action("action.dat");
}

```
bool movie::fileinput (ifstream & fin)
{
    bool success = false;
    fin.get(title, 131, ':');
    fin.get();
    if( fin && !fin.eof())
    {
        fin.get(review, 501, '\n');
        fin.get();
        success = true
    }
    return success;

}
```



Class interface

```
class library
{
    library();
    library(char[]);
    ~library();
};
```

NO return types

← destructor

↑ NO arg's

---

```
library::~library()
{
    ofstream fout;
    fout.open(stored_filename);
    for(int i=0;
    if (fout)
    {
```

```
        for (int i = 0; i < num_of_movies; ++i)
            database[i].fileoutput(fout);
        fout.close();
}

void movie::fileoutput(ofstream &fout)
{
    fout << title << ':' << review << endl;
}
```