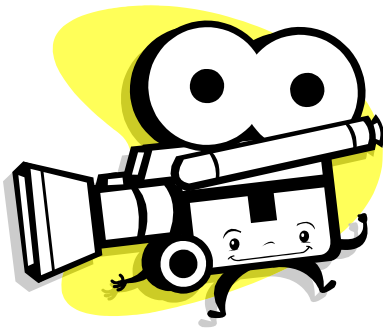


P7 – Movie Database Program



Initial Thoughts

- Each movie has:
 - a name (a **string**)
 - a rating (integer)

We can use a **record** to bundle these together.

- For the string, we will use a record of type **str** ... so we will have a record inside of a record!
- Furthermore: we need an **array** of movie records.

Declarations

```
type str = record
    sym: array[1..50] of char;
    len: integer
end;

Movie = record
    name: str;
    rating: integer
end;

Mlist = array[1..50] of Movie;

var movies: Mlist;
    n: integer;
    choice: integer;
    data: text;
```

Hint: Draw a picture of this! Very helpful!

Strings Code

- We just finished covering a group of procedures/functions for reading/writing strings.**
- You will need all of this code!**
- I will post it in on the web site, so that you can insert it into your programs.**

A Starting Point Program

- **Print Procedure**
just prints: **THE MOVIE DATABASE**
- **Menu Procedure (the usual!)**
 - prints options
 - gets choice from user
- **Main Program**

```
begin
    n := 0; { important! }
    repeat
        Print (movies, n);
        Menu (choice);
        case choice of
            1: ;
            2: ;
            3: ;
            4: ;
            5: ;
        end;
    until .....
end.
```

Adding a new movie: addMovie procedure

2 parameters:

- the movies array**
- the number of movies**

Assume n is the current number of movies.

First: $n := n + 1$;

Then:

write ('Enter the movie name: ');

strRead(_____);

write ('Enter your rating: ');

readln(movies[n]. rating);

Printing the Database

- 2 parameters needed:
 - the movies array
 - the number of movies
- If you were to print the info for just the first movie, it would look like this:

```
write( movies[1].rating:2 );
```

```
strWrite( _____ );
```

```
writeln; ← easy to forget
```

... but you will need to use a loop instead.

Clearing the Database

```
n := 0;
```

Loading The Database From the File (the load procedure)

- **Suggestions:**
 - **2 parameters:**
 - **the movies array**
 - **the number of movies (must be a *var* parameter!)**
 - **declare a file variable of type text**
- **First, don't forget about details like "assign" and "reset".**
- **Here, I assume data is the name of the file variable (or parameter)**
- **First, use "readln" to read the number of movies in the database into N.**

Please don't use "read."

- **Next, use a loop (1 to N) to read each title and rating into a record.**

If you were reading just 1 title, and 1 rating, it would look like this:

```
strReadf( data, _____ );  
readln( data, movies[1].rating );
```

(Again, please don't use "read" .)

.... so just modify this for a loop

Don't forget "close"

Saving the Database To the File (the save procedure)

- **Suggestions:**
 - 2 parameters:
 - 1 for the movies array
 - 1 for the number of movies
 - declare a file variable of type text
- First, don't forget about details like "assign" and "rewrite".
- Here, I assume data is the name of the file variable or parameter
- First, use `writeln` to write the number of movies into the file:


```
writeln( data, ..... );
```
- Next, use a loop (1 to N) to write each title and rating into the file.

If you were saving just 1 title, and 1 rating, it would look like this:

```
strWritef( data, _____ );  
writeln( data ); ← don't forget!  
writeln( data, movies[1].rating );
```

.... so just modify this for a loop

Also, don't forget about "close"

Sorting the Array of Movies

- much easier than you'd expect!
- start with your sort procedure from P5
- modify the parameter ... the type should be Mlist now
- Adjust the code to compare the ratings of individual elements of of the movie array.
Hint: the rating of the first movie is `movies[1].rating` ... so what is the rating of the `i`th movie, and the `i+1st` movie?
- For swapping:
 - Use a variable called `temp` of type `Movie`. (`temp` is a single `Movie` rec.)
 - Hint: remember that you can assign 1 record to another just by using a regular assignment statement!

Bubble Sort Procedure for a 1-D Array (parts that should be changed are in color)

```
procedure sort( var values: List );  
var pass, i : integer;  
    temp: integer;  
  
begin  
    for pass := 1 to 49 do  
        for i := 1 to 49 do  
            if values[ i ] < values[i+1] then  
                begin  
                    temp := values[ i ];  
                    values[ i ] := values[i+1];  
                    values[i+1] := temp  
                end;  
        end;  
    end;
```

Note: you will need a 2nd parameter.
Can you guess what it is?

Finishing Touches

- **Suggestion: wait until the end, to polish the output**
- **OK to add new features:**
 - **additional fields for each movie:**
 - **actor/actress (or multiple)**
 - **year**
 - **genre (comedy, action ...)**
 - **sort by title**
 - **find a particular movie**
 - **delete a movie (hint: number each movie in the output, so the user can specify which number to delete)**
 - **allow a short comment from the user for each movie**