Program Design Final Project-

Data Base

NBA Data Comparison

Team Member

407210046 許家和

408420001 王宇軒

408420069 陳韋丞

408420082 周韋良

409220042 黃國琨

* Introduction section

Background of this project:

When we are discussing the topic for the final project, some teammates talk about NBA has a lot of data to each player.

Since completing the final project with the application of database, we need a lot of data to implement it.

So, we have the idea to analyze and make some function to these data. In one purpose to complete the final project, and the other purpose is to deal with the data which we are interesting in.

Functions’ simple introduction:

There are mainly 9 functions we made for this project, listed sequentially below:

1.Add:

Add a datum into database.

2.Delete:

Delete a datum from databas.

3.Compare:

Compare each different item types between two data.

4.Search:

Print the target you want know about in the database.

5.Sort (in ascend):

Sorting depend on given item form in ascendance.

6.Sort (in descend):

Sorting depend on given item form in descendance.

7.Traverse:

Print out the database in two different ways.

8.Personal\_analysis:

Solve the confusion for the same player with many different team names or data.

9.Exit:

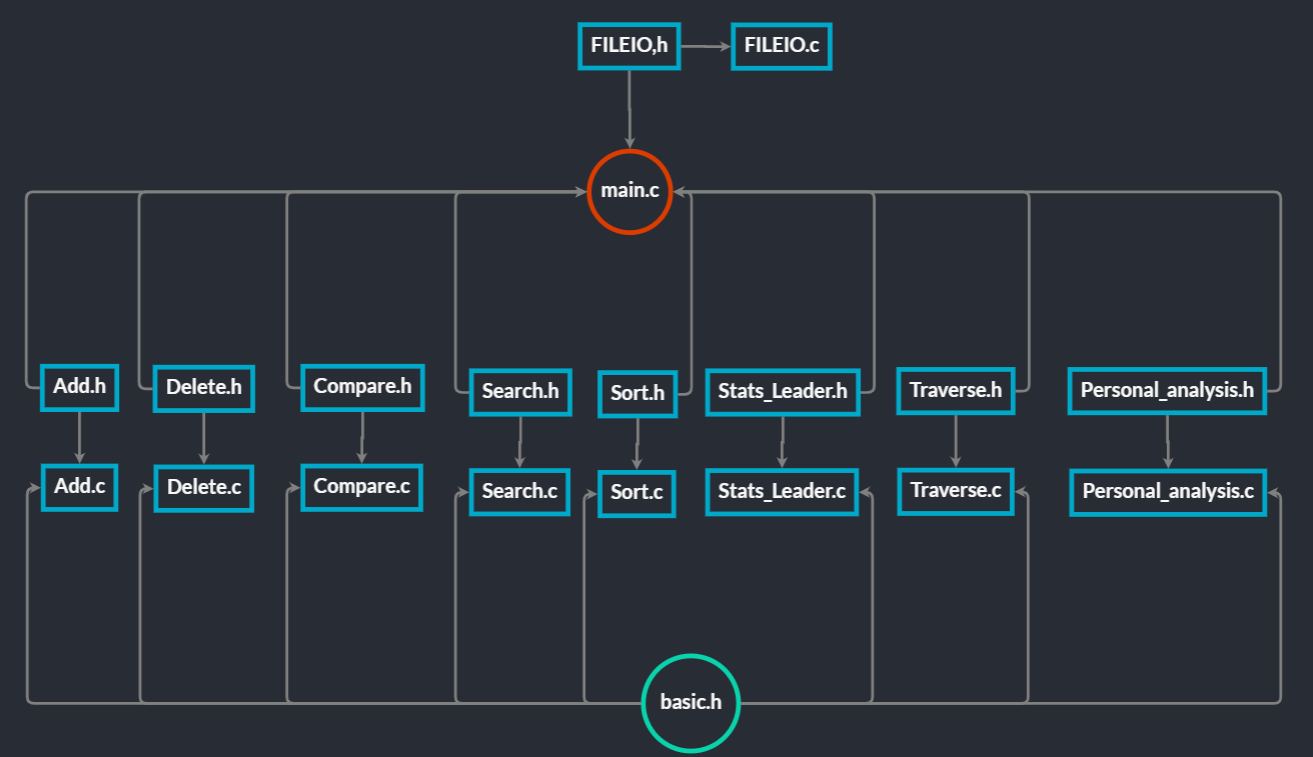
Leave and terminates the program.

* Program Design section

In this section, we are going to introduce the structure of our program and relationship between each files.

Due to the complicate relationship between all files, we cannot explain them in words, so we will present in the form of a diagram as below.

The arrow means the file is imported into the one which point to.



* Basic Part section

Data Type and Data Structure

1.at least basic data types:

We used int,float,char,pointer,structure,function

2.at least a string type

Use char[] as the string type

3.at least one of structure, union, or enumeration

We defined structure in file “basic.h”

4.link list for database

Read a txt file to form a database inlinked list form

Operation

(Add,Delete,Traverse,Search,Sort\*2,File I/O)

Add:Add a new item to the database system.

And the program must print an error message,

if the item is already in the database.

1.How to use

example: Add(list , new\_member);

2.Type

voidAdd(struct item \*, struct item \*);

Type function: void

Type parameters:Both are struct item \* (include from “basic.h”)

3.each means of parameters

voidAdd(struct item \*head, struct item \*new\_member){

    /\*(the former is the address point to the head of the database

    ,the latter is who you want to add into)\*/

    struct item \*origin;

    origin = head;       // record the original place

4.Function Main Body 1:secure valid input

if (head ==NULL) {  // check if the list is empty

        printf("the list is empty.\n");

        return;

    }

5.Function Main Body 2:secure no repeat players

Use for loop traversing every each member, then use strcmp to identify if total\_name and team is the same to the target.

If the item is already in the database, print the message "the item is already in the database.".

Then return, refrain from doing the following code behind.

for (head=origin; head !=NULL; head =head->next) {

        if (strcmp(new\_member->total\_name, head->total\_name)==0

&&strcmp(new\_member->team, head->team)==0) {

            head = origin;

            printf("the item is already in the database.\n");

            return;

        }

    }

6.Function Main Body 3:add to the last one of the database

Use for loop until encountering NULL, add the new\_member

for (head=origin; head !=NULL; head =head->next) {

        if (head->next==NULL) {             // add to the last

            head->next= new\_member;

            head = origin;

            return;

        }

    }

}

Delete:Delete a given item from the database.

And the program must print an error message,

if there is no matching item to be deleted in the database.

1.How to use

example: list=Delete(list,"target\_name");

Then scanf the target team.

2.Type

struct item \*Delete(struct item \*head, char\*Name)

Type function: struct item \*(include from “basic.h”)

Type parameters:struct item \* and char \*

3.each means of parameters

struct item \*Delete(struct item \*head, char\*Name) {

    /\*(the former is the address point to the head of the database

    ,the latter is who you want to delete)\*/

    struct item \*origin;

    charTeam[32];//the target team

    origin = head;       // record the original place

4.Function Main Body 1: secure valid input

if (head ==NULL) {  // check if the list is empty

        printf("the list is empty.\n");

        returnorigin;

    }

5.Function Main body 2:use search function to show the same names in teams

show the same names in teams,

Then,ask for the target team in which the member is

Search(&head, "NAME", Name);      //Firstly print the same names in teams

    printf("which team :");

    scanf("%s", Team);                     //Then enter  the target team

6.Function Main Body 3:delete the target under 3 conditions if the item is already in the database.

Condition I:delete the first one and return brand new head.

if (strcmp(Name, origin->total\_name)==0&&strcmp(Team, origin->team)==0) {

    struct item \*temp;

    temp =head->next;

    free(head);

    head =head->next;

    return head;

    }

Condition II & III:delete the middle one & delete the last one

Use for loop traversing every each member, and then stop at the previous one which target is the next one to strcmp() determine if next->total\_name &next->team is the same to target.

Declare another list for target->next

Use free() to delete

If delete the middle one, connect with temp

Then return,avoid to print the message about no matching player to delete.

    for (head = origin; head !=NULL; head =head->next) {

        if (strcmp(Name, head->next->total\_name) ==0&&strcmp(Team, head->next->team) ==0) {

            struct item \*temp;

            if (head->next->next!=NULL) {  // delete the middle one

                temp =head->next->next;

                free(head->next);

                head->next= temp;

            } else {  // delete the last one

                free(head->next);

                head->next=NULL;

            }

            returnorigin;

        }

    }

Condition IV: No matching player to delete and return

printf("there is no matching item to be deleted in the database.\n");

returnorigin;

}

Sort\*2: sort ascend or descend data which you choose

1.How to use

example: Sort\_ascending(&list , PTS),Sort\_descending(&list , FG)

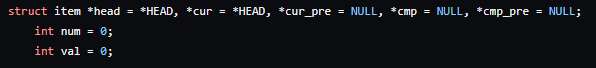
2.Type



Type function: void

Type parameters: First is struct item \* (include from “basic.h”),the other is string

3.each means of parameters



head, cur: address point to the head of database

cur\_pre:address point to the cur’sprevious node

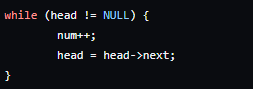
cmp: address point to node compare to cur

cmp\_pre:address point to the cmp’sprevious node

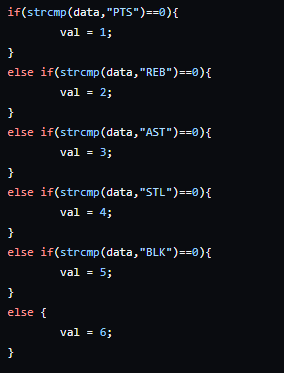
val: choose which datayou want to sort

num: calculate the time you sort

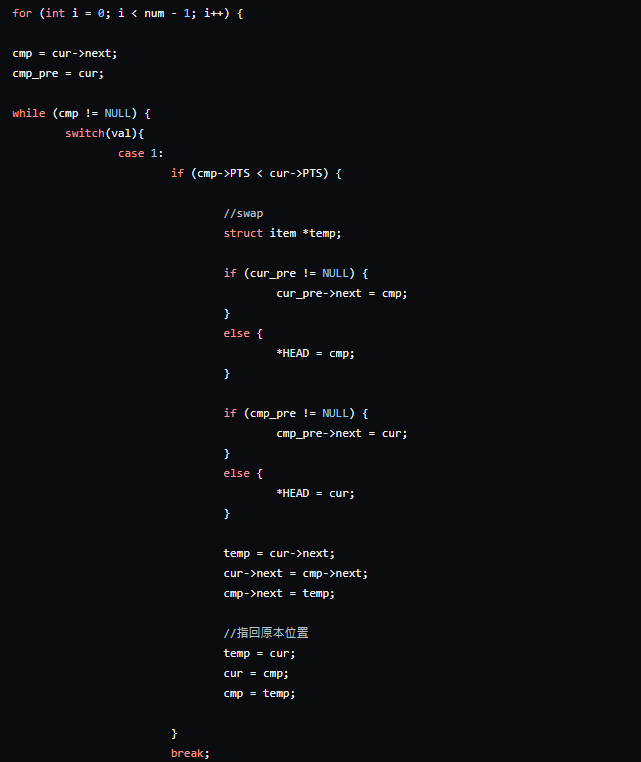
4.Function Main Body 1:calculate amounts of player for sorting

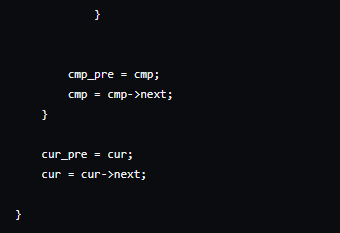


5.Function Main Body 2:give each datum a number for switch



6.Function Main Body 3:error message





Use for loop get the “first” item of list and start comparing to forward, exchange when meeting smaller one until the end of list.

Then get “second” item of list, and so on, until the front item of list.

(find smallest one from “n”, and then get smallest one from”n-1”, and so on.)

Serach:Given certain information about the item, the program needs to find and print the specific item.

1.How to use

Example:

search\_list(&list, "NAME", "Stephen Curry");

search\_list(&list, "TEAM", "LAL");

search\_list(&list, "POS", "C");

2.Type

voidSearch(structitem \*\*HEAD, char \*type, char \*goal)

function type: void

function parameters: linked-list and two strings

3.each means of parameters

intsearch = 0;

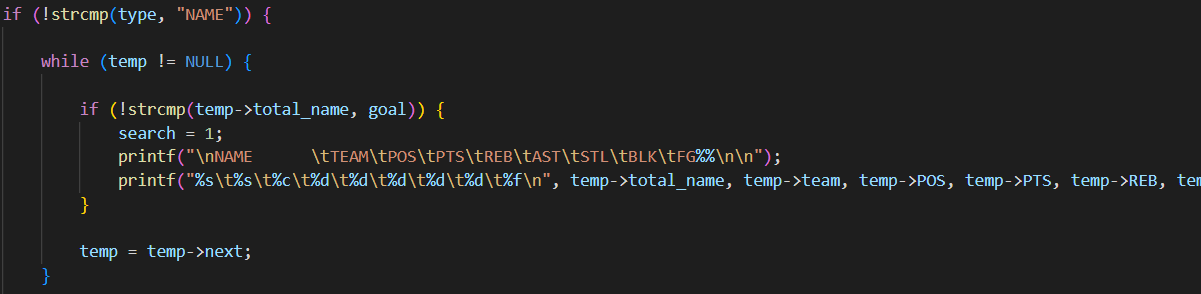
structitem \*temp = \*HEAD;

search: record if the target has been find

temp: store head of list temporary

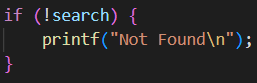
4.

Function Main Body 1



Determine item types(NAME ,TEAM, POS), and then traverse from head, if found let search equals to 1 and print out all target data.

Function Main Body 2



If search = 0, print “Not Found”

Traverse:Print all items in the database in a specific format.

1. How to use:

Example: Traverse(&list);

1. Type:

voidTraverse(structitem \*\*HEAD)

function type: void

function parameter: linked-list

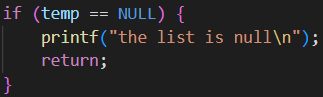
1. each means of parameter

structitem \*temp = \*HEAD;

temp: store head of list for temporary

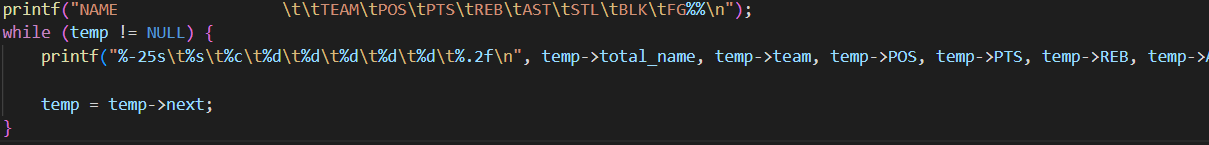
4.

FunctionMain Body 1



If list is empty, report error and return nothing.

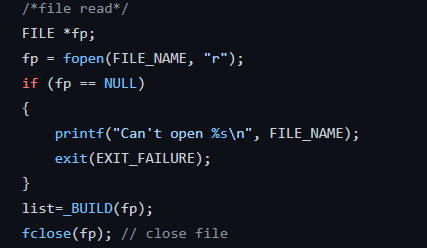
FunctionMain Body 2



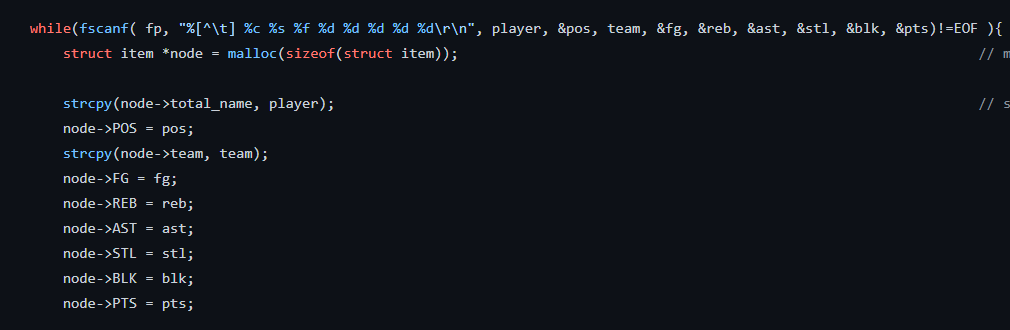
Print out from head in certain form.

File IO:read data from txt file, and then store it in linked list form, rewrite the txt file after termination of the program.

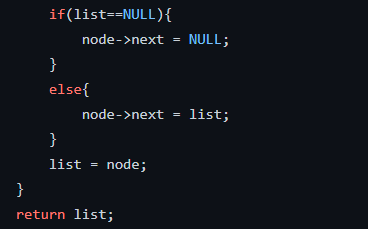
Open and read txt file, use \_BUILD function to make linked list, then close txt file after linked list construction.



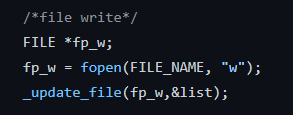
\_BUILD function:read in data from txt file until EOF, malloc a struct for each rowand store them.



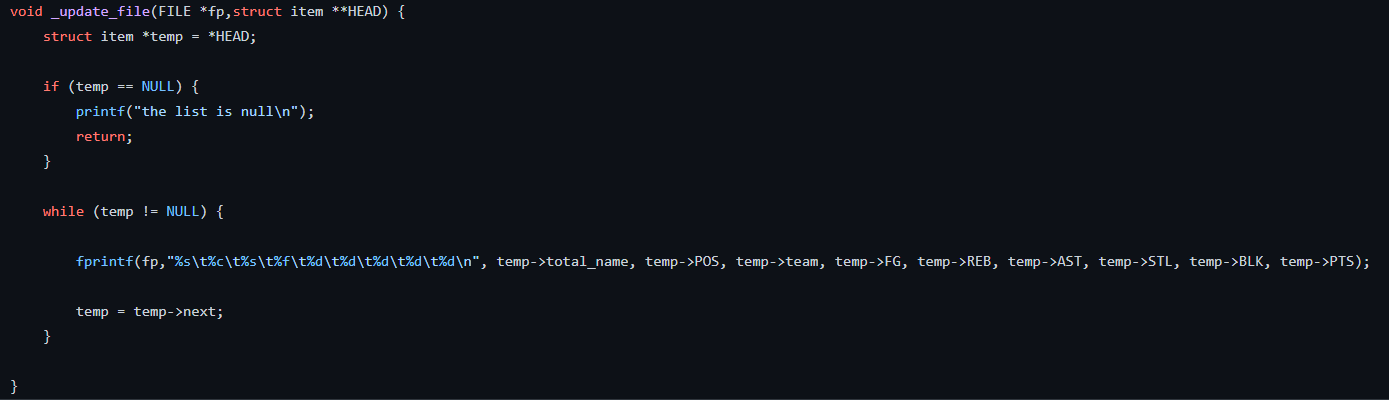
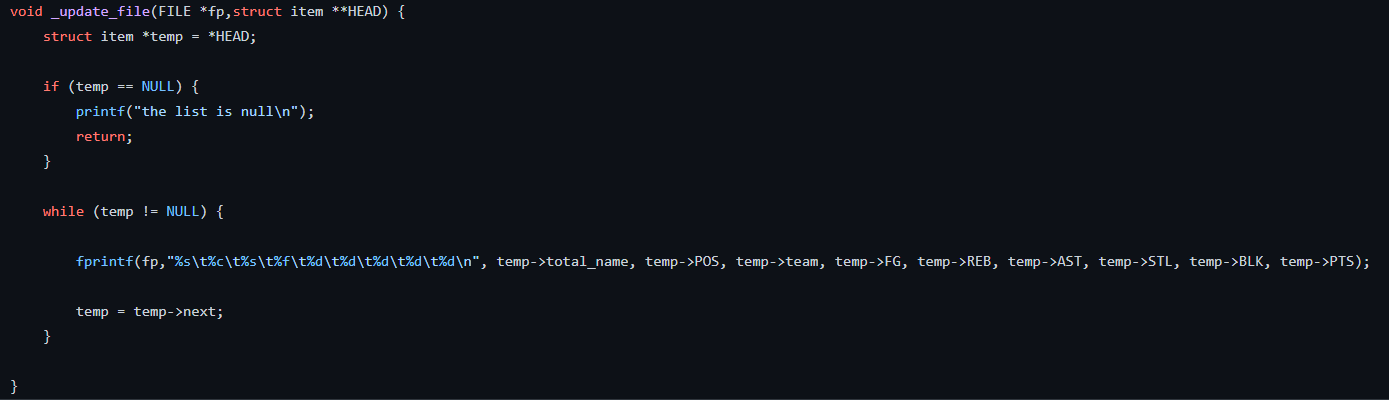
Link new structure to list, then return list to main



Open txt file, then use \_update\_fileto write data in.



\_update\_file function:use fprintf to write data into txt file



* Advance Part section

1. Use advance data structure(not done)

2. Implement the searching or sorting algorithms based on the advanced data structures.(not done)

3. Interface(not done)

4. Useful extra functions

Compare: compare two players data.

And the program must print an error message,

If the player doesn’t exited in the database.

1.How to use

example: Compare(list , player1,player2);

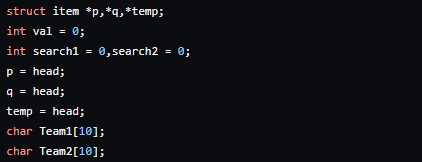
2.Type



Type function: void

Type parameters: First is struct item \* (include from “basic.h”),the other two are string

3.each means of parameters



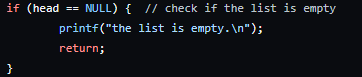
p,q, temp: address point to the head of database, which used to find player you want.

val: check the player exit in database

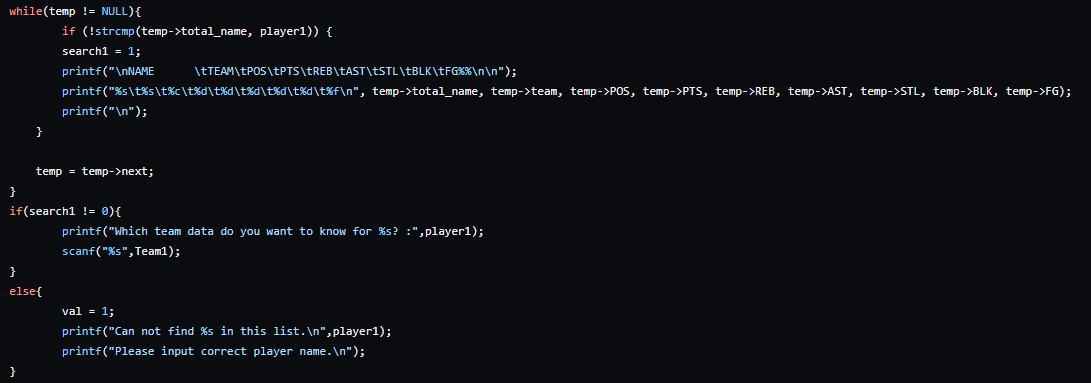
search1, search2 : check the player1, player2 have be found in database

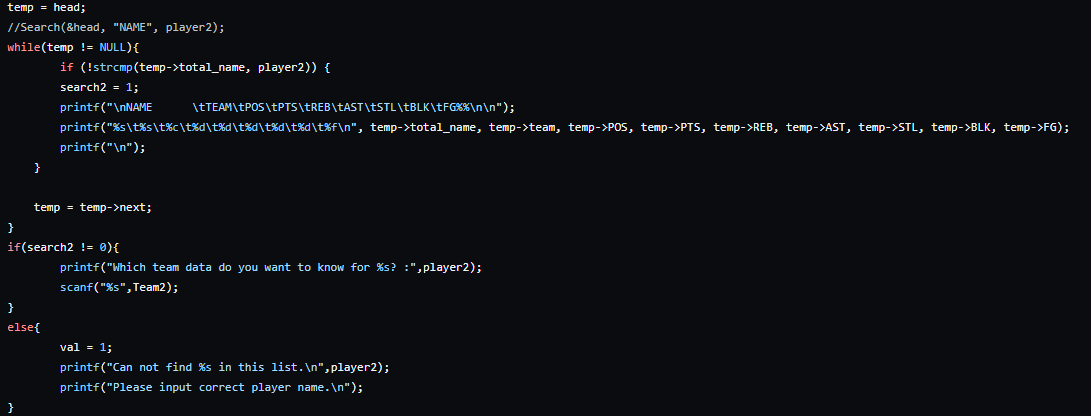
Team1, Team2: select the team you want to compare(player season Transfers)

4.Function Main Body 1:secure valid input



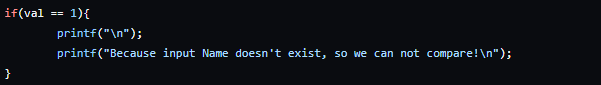
5.Function Main Body 2:find player1&player2



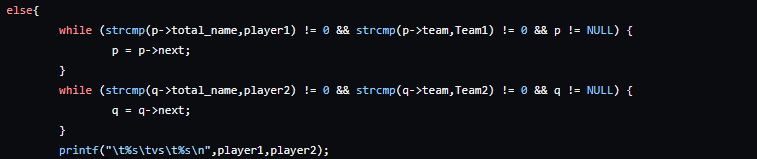


Use while loop searching target player, if found(search1 or search2 equals to 1), choose a team for player to compare

6.Function Main Body 3:error message

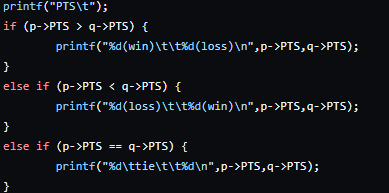


7.Function Main Body 4:find two players



Use while loop searching target name, print out players

8.Function Main Body 5:compare two players



Compare two player depend on item type(PTS,BLK,AST,STL,FG,REB), then print out who is better.

Personal\_analysis:Sort and Print the player in many teams.

If you are curious about the specific player,

and try to compare some datas between the different teams.

Beside,it will only show datas personally.

1. How to use:

Example: Personal\_analysis(list,the\_player’s\_name);

1. Type

The main function:

voidPersonal\_analysis(struct item \*, char\*);

Type function: void

Type parameters:struct item \* (include from “basic.h”) and char \*

The custom function:(check malloc() is safe)

voidbuild(struct item \*\*)

Type function: void

Type parameters:struct item \*\*

1. each means of parameters

The main part:

voidPersonal\_analysis(struct item \*head, char\*Name){

struct item \*temp;//暫存用

struct item\*personal\_datas;/\*point to the head of the new link lists

for same player in many teams \*/

personal\_datas =NULL;//initialize

struct item\*original; // record the address of the head of personal\_datas

    int count=0;// record the number of member of personal\_datas

In a if loop about building a new link lists for same player in many teams:

struct item \*new\_datas;// new member of personal\_datas

voidbuild(struct item \*\*new\_datas);

build(&new\_datas);

1. Function Main Bodt 1:building a new link lists for same player in many teams

Use while loop traversing every each member and use strcmp determine if total\_name are the same with target.

Then use original record the last one data(head of personal\_datas)

while (head!=NULL){

        if (!strcmp(head->total\_name, Name)) {

            struct item \*new\_datas;

            build(&new\_datas);

            strcpy(new\_datas->total\_name,head->total\_name);

            strcpy(new\_datas->team,head->team);

            new\_datas->POS=head->POS;

            new\_datas->PTS=head->PTS;

            new\_datas->REB=head->REB;

            new\_datas->AST=head->AST;

            new\_datas->STL=head->STL;

            new\_datas->BLK=head->BLK;

            new\_datas->FG=head->FG;

            original=new\_datas;

            new\_datas->next= personal\_datas;

            personal\_datas = new\_datas;

            count++;

        }

        head =head->next;

    }

personal\_datas=original;

1. FunctionMain Body 2:use count determine amount of teams

Condition I:less than two teams to analyze than print the message.

if(count==0){

        printf("there is no matching player to be analyzed in the database.\n");

    }

    elseif(count==1){

        printf("the player could not be analyzed in only one team.\n");

    }

Condition II:if this player has played for more than one team

elseif(count>1){

        charcmp\_data[10];// the data you want to compare

        int opt=0;// the option to decide the Sort function to ascend or descend

        printf("input PTS/AST/BLK/STL/REB/FG:");

Firstly, scanf and check the right cmp\_data.

scanf("%s", cmp\_data);

while(strcmp(cmp\_data,"PTS")&&strcmp(cmp\_data,"AST")&&strcmp(cmp\_data,"BLK")

&&strcmp(cmp\_data,"STL")&&strcmp(cmp\_data,"REB")&&strcmp(cmp\_data,"FG")){

            printf("input PTS/AST/BLK/STL/REB/FG:");

            scanf("%s", cmp\_data);

        }

Simultaneously,scanf and check the right option.

printf("[1]Sort(ascend) [2]Sort(descend):");

        scanf("%d", &opt);

        while(opt!=1&&opt!=2){

            printf("[1]Sort(ascend) [2]Sort(descend):");

            scanf("%d", &opt);

        }

Finally,use the sort function and Traverse function to show the personal\_datas designed by user.

if(opt==1){

            Sort\_ascending(&personal\_datas, cmp\_data);

            Traverse(&personal\_datas);

        }

        if(opt==2){

            Sort\_descending(&personal\_datas, cmp\_data);

            Traverse(&personal\_datas);

        }

    }

}

State\_leader function: print all data top5 player and print season leader.

1.How to use

example: Stats\_leader(&list);

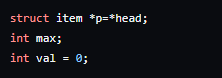
2.Type



Type function: void

Type parameters: struct item \* (include from “basic.h”)

3.each means of parameters

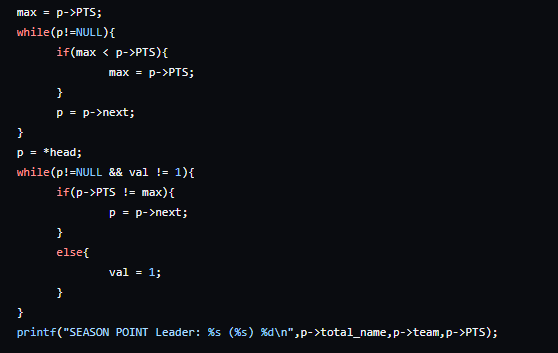


p: address point to the head of database

max: data maximum

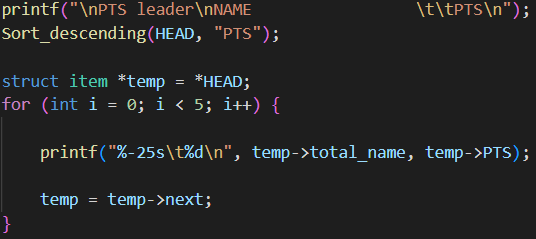
val: parameter used for stop searching

4.Function Main Body 2: search and print highest point player



Use while loop search highest PTS, then print the player with it, and so on.

5.functionMain Body 2 (find the five highest player for each item type)



Sorting with sort and print five highest in each item.

* Demonstration section

Github repository : https://github.com/v0103/final.git

\*Choose branch ”fp”