

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:

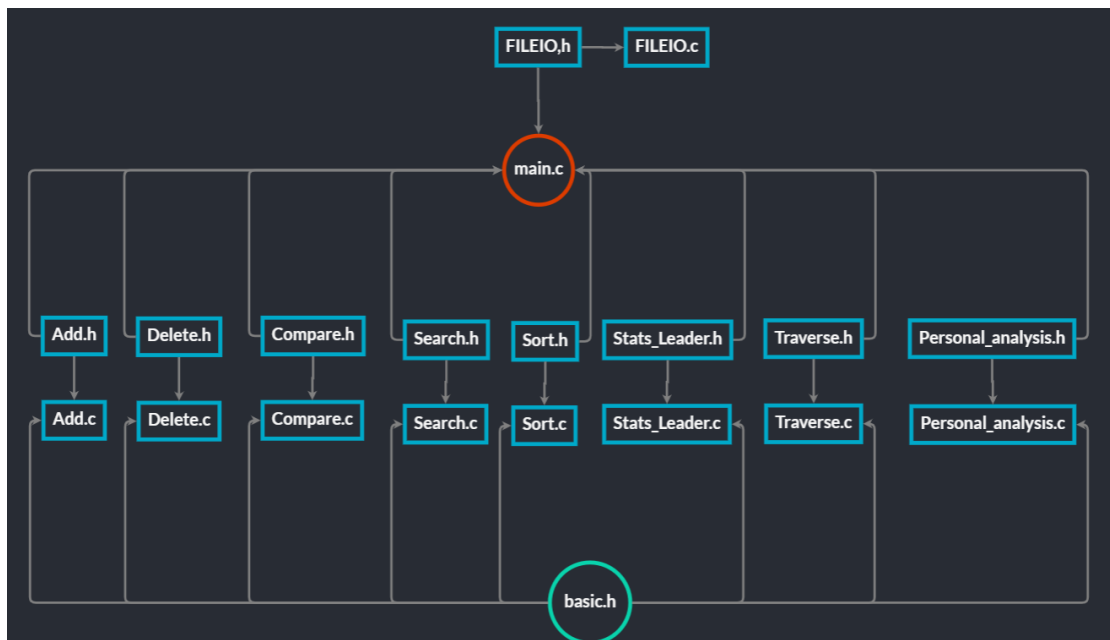
Leave and terminates the program.

- 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 in linked 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

```
void Add(struct item *, struct item *);
```

Type function: void

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

3.each means of parameters

```
void Add(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

```
void Sort_ascending(struct item **HEAD, char data[10]) {
```

Type function: void

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

3.each means of parameters

```
struct item *head = *HEAD, *cur = *HEAD, *cur_pre = NULL, *cmp = NULL, *cmp_pre = NULL;  
int num = 0;  
int val = 0;
```

head, cur: address point to the head of database

cur_pre: address point to the cur's previous node

cmp: address point to node compare to cur

cmp_pre: address point to the cmp's previous node

val: choose which data you want to sort

num: calculate the time you sort

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

```
while (head != NULL) {  
    num++;  
    head = head->next;  
}
```

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

```
if(strcmp(data,"PTS")==0){  
    val = 1;  
}  
else if(strcmp(data,"REB")==0){  
    val = 2;  
}  
else if(strcmp(data,"AST")==0){  
    val = 3;  
}  
else if(strcmp(data,"STL")==0){  
    val = 4;  
}  
else if(strcmp(data,"BLK")==0){  
    val = 5;  
}  
else {  
    val = 6;  
}
```

6. Function Main Body 3: error message

```

for (int i = 0; i < num - 1; i++) {

    cmp = cur->next;
    cmp_pre = cur;

    while (cmp != NULL) {
        switch(val){
            case 1:
                if (cmp->PTS < cur->PTS) {

                    //swap
                    struct item *temp;

                    if (cur_pre != NULL) {
                        cur_pre->next = cmp;
                    }
                    else {
                        *HEAD = cmp;
                    }

                    if (cmp_pre != NULL) {
                        cmp_pre->next = cur;
                    }
                    else {
                        *HEAD = cur;
                    }

                    temp = cur->next;
                    cur->next = cmp->next;
                    cmp->next = temp;

                    //指回原本位置
                    temp = cur;
                    cur = cmp;
                    cmp = temp;

                }
                break;

```

```

        }

        cmp_pre = cmp;
        cmp = cmp->next;
    }

    cur_pre = cur;
    cur = cur->next;
}

```

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.)

Search: 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

```
void search(struct titem **HEAD, char *type, char *goal)
```

function type: void

function parameters: linked-list and two strings

3. Each means of parameters

```
int search = 0;
struct titem *temp = *HEAD;
```

search: record if the target has been found

temp: store head of list temporary

4.

Function Main Body 1

```
if (!strcmp(type, "NAME")) {
    while (temp != NULL) {
        if (!strcmp(temp->total_name, goal)) {
            search = 1;
            printf("\nNAME\tTEAM\tPOS\tPTS\tREB\tAST\tSTL\tBLK\tFG%%\n\n");
            printf("%s\t%s\t%c\t%d\t%d\t%d\t%d\t%d\t%f\n", temp->total_name, temp->team, temp->POS, temp->PTS, temp->REB, temp->AST, temp->STL, temp->BLK, temp->FG);
        }
        temp = temp->next;
    }
}
```

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) {
    printf("Not Found\n");
}
```

If search = 0, print "Not Found"

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

1. How to use:

Example: Traverse(&list);

2. Type:

```
void Traverse(structitem **HEAD)
```

function type: void

function parameter: linked-list

3. each means of parameter

```
structitem *temp = *HEAD;
```

temp: store head of list for temporary

4.

FunctionMain Body 1

```
if (temp == NULL) {  
    printf("the list is null\n");  
    return;  
}
```

If list is empty, report error and return nothing.

FunctionMain Body 2

```
printf("NAME\t\tTEAM\tpos\tpTS\tREB\tAST\tSTL\tBLK\tFG%%\n");  
while (temp != NULL) {  
    printf("%-25s\t%s\t%c\t%d\t%d\t%d\t%d\t%d\t%.2f\n", temp->total_name, temp->team, temp->pos, temp->PTS, temp->REB, temp->AST, temp->STL, temp->BLK, temp->FG);  
    temp = temp->next;  
}
```

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.

```

/*file read*/
FILE *fp;
fp = fopen(FILE_NAME, "r");
if (fp == NULL)
{
    printf("Can't open %s\n", FILE_NAME);
    exit(EXIT_FAILURE);
}
list=_BUILD(fp);
fclose(fp); // close file

```

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

```

while(fscanf( fp, "%[^\t] %c %s %f %d %d %d %d %d\n", player, &pos, team, &fg, &reb, &ast, &stl, &blk, &pts)!=EOF ){
    struct item *node = malloc(sizeof(struct item)); // m
    strcpy(node->total_name, player); // s
    node->POS = pos;
    strcpy(node->team, team);
    node->FG = fg;
    node->REB = reb;
    node->AST = ast;
    node->STL = stl;
    node->BLK = blk;
    node->PTS = pts;
}

```

Link new structure to list, then return list to main

```

if(list==NULL){
    node->next = NULL;
}
else{
    node->next = list;
}
list = node;
}
return list;

```

Open txt file, then use _update_file to write data in.

```

/*file write*/
FILE *fp_w;
fp_w = fopen(FILE_NAME, "w");
_update_file(fp_w,&list);

```

_update_file function:use fprintf to write data into txt file

```
void _update_file(FILE *fp,struct item **HEAD) {
    struct item *temp = *HEAD;

    if (temp == NULL) {
        printf("the list is null\n");
        return;
    }

    while (temp != NULL) {

        fprintf(fp,"%s\t%c\t%s\t%f\t%d\t%d\t%d\t%d\t%d\t%d\n", temp->total_name, temp->POS, temp->team, temp->FG, temp->REB, temp->AST, temp->STL, temp->BLK, temp->PTS);

        temp = temp->next;
    }
}
```

```
\t%d\n", temp->total_name, temp->POS, temp->team, temp->FG, temp->REB, temp->AST, temp->STL, temp->BLK, temp->PTS);
```

● 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 exist in the database.

1.How to use

example: Compare(list , player1,player2);

2.Type

```
void Compare(struct item *head, char *player1, char *player2)
```

Type function: void

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

3.each means of parameters

```
struct item *p,*q,*temp;  
int val = 0;  
int search1 = 0,search2 = 0;  
p = head;  
q = head;  
temp = head;  
char Team1[10];  
char Team2[10];
```

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

val: check the player exist 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


```

if (head == NULL) { // check if the list is empty
    printf("the list is empty.\n");
    return;
}

```

5.Function Main Body 2:find player1&player2

```

while(temp != NULL){
    if (strcmp(temp->total_name, player1)) {
        search1 = 1;
        printf("\nNAME      \tTEAM\tPOS\tPTS\tREB\tAST\tSTL\tBLK\tFG%\n\n");
        printf("%s\t%s\t%c\t%d\t%d\t%d\t%d\t%d\t%f\n", temp->total_name, temp->team, temp->POS, temp->PTS, temp->REB, temp->AST, temp->STL, temp->BLK, temp->FG);
        printf("\n");
    }

    temp = temp->next;
}

if(search1 != 0){
    printf("Which team data do you want to know for %s? :",player1);
    scanf("%s",Team1);
}
else{
    val = 1;
    printf("Can not find %s in this list.\n",player1);
    printf("Please input correct player name.\n");
}

```

```

temp = head;
//Search(&head, "NAME", player2);
while(temp != NULL){
    if (strcmp(temp->total_name, player2)) {
        search2 = 1;
        printf("\nNAME      \tTEAM\tPOS\tPTS\tREB\tAST\tSTL\tBLK\tFG%\n\n");
        printf("%s\t%s\t%c\t%d\t%d\t%d\t%d\t%d\t%f\n", temp->total_name, temp->team, temp->POS, temp->PTS, temp->REB, temp->AST, temp->STL, temp->BLK, temp->FG);
        printf("\n");
    }

    temp = temp->next;
}

if(search2 != 0){
    printf("Which team data do you want to know for %s? :",player2);
    scanf("%s",Team2);
}
else{
    val = 1;
    printf("Can not find %s in this list.\n",player2);
    printf("Please input correct player name.\n");
}

```

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

```

if(val == 1){
    printf("\n");
    printf("Because input Name doesn't exist, so we can not compare!\n");
}

```

7.Function Main Body 4:find two players

```

else{
    while (strcmp(p->total_name,player1) != 0 && strcmp(p->team,Team1) != 0 && p != NULL) {
        p = p->next;
    }

    while (strcmp(q->total_name,player2) != 0 && strcmp(q->team,Team2) != 0 && q != NULL) {
        q = q->next;
    }

    printf("\t%s\tvs\t%s\n",player1,player2);
}

```

Use while loop searching target name, print out players

8.Function Main Body 5:compare two players

```
printf("PTS\t");
if (p->PTS > q->PTS) {
    printf("%d(win)\t\t%d(loss)\n",p->PTS,q->PTS);
}
else if (p->PTS < q->PTS) {
    printf("%d(loss)\t\t%d(win)\n",p->PTS,q->PTS);
}
else if (p->PTS == q->PTS) {
    printf("%d\ttie\t\t%d\n",p->PTS,q->PTS);
}
```

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);

2.Type

The main function:

```
void Personal_analysis(struct item *, char*);
```

Type function: void

Type parameters:struct item * (include from "basic.h") and char *

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

```
void build(struct item **)
```

Type function: void

Type parameters:struct item **

3.each means of parameters

The main part:

```
void Personal_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

void build(struct item **new_datas);

build(&new_datas);
```

4.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;
```

5.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){
```

```

char cmp_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

```
void Stats_leader(struct item **HEAD)
```

Type function: void

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

3. each means of parameters

```
struct item *p=*head;
int max;
int val = 0;
```

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

```
max = p->PTS;
while(p!=NULL){
    if(max < p->PTS){
        max = p->PTS;
    }
    p = p->next;
}
p = *head;
while(p!=NULL && val != 1){
    if(p->PTS != max){
        p = p->next;
    }
    else{
        val = 1;
    }
}
printf("SEASON POINT Leader: %s (%s) %d\n",p->total_name,p->team,p->PTS);
```

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)

```
printf("\nPTS leader\nNAME\t\tPTS\n");
Sort_descending(HEAD, "PTS");

struct item *temp = *HEAD;
for (int i = 0; i < 5; i++) {

    printf("%-25s\t%d\n", temp->total_name, temp->PTS);

    temp = temp->next;
}
```

Sorting with sort and print five highest in each item.

- Program function instructions(github,README)

Introduction

How to use the program

*suggestion

you may print out the database first with
function[7]:Traverse and choose [1] to see all of
the database first*

enter option 1~9 to execute the program

1.Add:Add a new item to the database system.

input 1, then input each item type for a
player

2.Delete>Delete a given item from the database.

input 2, then input a player's name, and
then choose a team for the player

3.Compare: compare two players data.

input 3, then input first player's name,
then input second player's name
then input the correct team for
player1,player2

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

input 4, then input item
type(NAME,TEAM,POS), then input datum for
the item type

5.Sort(ascend):sort data in ascend form based on chosen item type

input 5, then input a item type

6.Sort(descnd):sort data in descend form basd on chosen item type

inpur 6, then input a item type

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

input 7,then input 1 for all database;2 for bestplayer and five highest for each item type

8.Personal_analysis:Sort and Print the player in many teams.

input 8,then input player's name,then
input item type,and then choose ascend or
descend form for data

9.Exit:terminate the program

input 9

- Demonstration section

Github repository :

<https://github.com/v0103/final.git>

*Choose branch "fp"