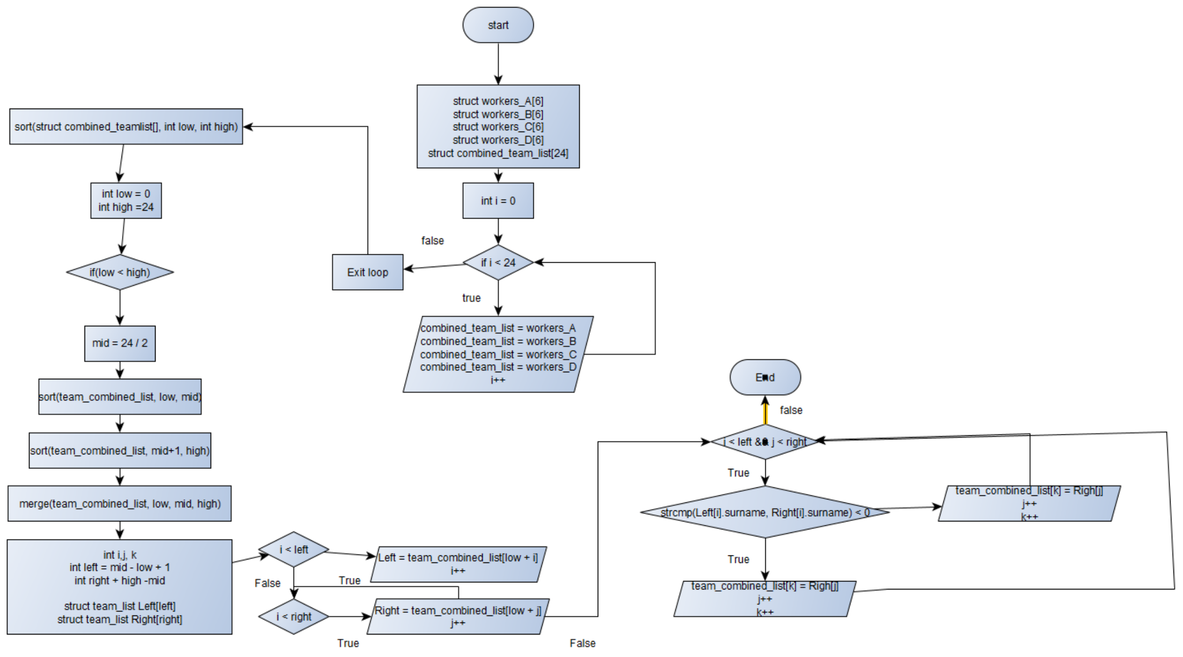
Algorithms & Design Assignment

Deyor Abidjanov, C21446556

Flowchart for the Combine & Sort process – part 1:


start
struct workers_A[6]
struct workers_B[6]
struct workers_C[6]
struct workers_D[6]
struct combined_team_list[24]
int i = 0
if i < 24
false
Exit loop
true
combined_team_list = workers_A
combined_team_list = workers_B
combined_team_list = workers_C
combined_team_list = workers_D
i++
sort(struct combined_teamlist[], int low, int high)
int low = 0
int high =24
if(low < high)
mid = 24 / 2
sort(team_combined_list, low, mid)
sort(team_combined_list, mid+1, high)
merge(team_combined_list, low, mid, high)
int i,j, k
int left = mid - low + 1
int right + high -mid
struct team_list Left[left]
struct team_list Right[right]
i < left
Left = team_combined_list[low + i]
i++
False
True
i < right
Right = team_combined_list[low + j]
j++
True
False
End
false
i < left && j < right
True
strcmp(Left[i].surname, Right[i].surname) < 0
team_combined_list[k] = Righ[j]
j++
k++
True
team_combined_list[k] = Righ[j]
j++
k++

- To combine and sort the list in alphabetical order, I used the “Merge Sort” algorithm. This algorithm divides the array into two sub arrays. It sorts the first half of array, then moves on and sorts the second half. Once both halves are sorts, they are then merged again.
- As this is a recursive algorithm, the time complexity is logarithmic time, O(nlogn).

Pseudocode for Fully Certified List – part 2:

```
int main()
    call "fully_certified" function


void fully_certified
    int i
    FOR (i = 0; i < 24; i++)
        IF(combined_team_list[i].cert.cert_line == 3)
            PRINT( %s %s has certs in all 3 lines,
            combined_team_list[i].firstname,
            combined_team_list[i].surname)
        END
    END
```

- To form the list of the fully certified workers, I used a FOR loop with an IF statement. Therefore, the time complexity is Linear time, O(n)

Pseudocode for Search by Surname – part 3:

```
int main()
        char surname_search[12]
        PRINT (Enter Surname)
        SCAN (surname_seach)
        calling “search” function
        int search_result = search(surname_search)
        IF (search_result == - 1)
                PRINT (No Employee Found)
        ELSE
                PRINT (Employee’s ID: %d, search_result)


Int search(char lastname[12];
        Int i
        FOR ( i = 0; I < 24; i++)
                IF( STRING COMPARE (combined_team_list[i].surname to
                lastname))
                        RETURN    combined_team_list[i].cert.employee_ID
                END
        END
```

- To form the list of the fully certified workers, I used a FOR loop with an IF statement. Therefore, the time complexity is Linear time, O(n)

C Code for Program – part 4:

```c
/*
Program Description:

(a) This program combines the 4 team lists of a pharmaceutical company. Each
list is contained in an array of structures, and are all combined
    into an array of structures that's able to fit all the structure
variables. The team data structure will need at least the following items of
data - first name,
    surname, line number. After the lists are combined, the program is able to
produce a report of all the employees in surname order, alphabetically.
    To achieve this, the program uses the "Merge Sort" algorithm.

(b) The program can also produce a list all of the employees certified to work
on all 3 production lines in the company.
    The certification data structure will have employee id and earned
certification line id. This data structure is nested in the array of
structures use for team lists

(c) Finally, the program can execute a routine to search for a specific worker
by surname. The user is asked to enter the surname, and is then returned with
that
    employee's ID

Author: Deyor Abidjanov, C21446556

Date: 22/04/2022
*/

#include <stdio.h>
#include <string.h>

#define SIZE 6 // size of how many workers in each team list
#define NAME 12 // size of the array for names of employees
#define COMB 24 // size of total works in combined team list


//Structure templates

// structure nested in structure below
struct certifications
{
    int employee_ID;
    int cert_line_ID;
};

struct team_list
{
    char surname[NAME];
    char firstname[NAME];
    char team;
    int lineNumber;
    struct certifications cert; // nested structure
};


//Structure to store all team lists
struct team_list combined_team_list[COMB];


//Function Signatures

void sort(struct team_list array[], int low, int high);
void merge(struct team_list array[], int low, int mid, int high);
void fully_certified();
int search(char lastname[NAME]);


int main()
{
    int i;

    //first team list
    struct team_list workers_A[SIZE] = {
                        {"Obama", "Barack",'A', 1, 7, 2},
                        {"Jackson", "Michael", 'A', 1, 12, 2},
                        {"Parks", "Rosa", 'A', 3, 2, 2},
                        {"Gosling", "Ryan", 'A', 2, 1, 3},
                        {"Varadker", "Leo", 'A', 2, 99, 3},
                        {"Higgins", "Lucy",'A', 1, 123, 1}
                 };

    //second team list
    struct team_list workers_B[SIZE] = {
                        {"Streep", "Meryl",'B', 2, 34, 3},
                        {"Robinson", "Mary", 'B', 2, 21, 3},
                        {"Pitt", "Brad", 'B', 2, 87, 1},
                        {"Lee", "Bruce", 'B', 2, 102, 1},
                        {"Rousey", "Ronda",'B', 3, 9, 2},
                        {"Alba", "Jessica",'B', 1, 98, 2}
                 };

    // third team list
    struct team_list workers_C[SIZE] = {
                        {"Abidjanov", "Deyor", 'C', 2, 11, 2},
                        {"Tyson", "Mike", 'C', 1, 6, 1},
                        {"Stone","Emma", 'C', 2, 12, 2},
                        {"Chappelle", "Dave",'C', 2, 105, 2},
                        {"Martin", "Steve",'C', 2, 88, 2},
                        {"Smith", "Roger", 'C', 3, 65, 2}

                 };

    // fourth team list
    struct team_list workers_D[SIZE] = {
                        {"Nunes","Amanda", 'D', 3, 85, 3},
                        {"Murphy","Eddie", 'D', 90, 2},
                        {"Chan","Jackie",'D', 3, 3, 2},
                        {"Jackson", "Ian", 'D', 2, 111, 1},
                        {"Evans", "Lily", 'D', 2, 54, 2},
                        {"Bishop", "Lucy", 'D', 2, 77, 2}
                 };


    // Combining team lists into one list

    for(i = 0; i < SIZE; i++)
    {
        combined_team_list[i] = workers_A[i];
        combined_team_list[i + 6] = workers_B[i];
        combined_team_list[i + 12] = workers_C[i];
        combined_team_list[i + 18] = workers_D[i];
    }

    printf("\nUnsorted List of Employees:\n\n");

    for (i = 0; i < COMB; i++)
    {
        printf("Name: %s %s\n", combined_team_list[i].firstname,
combined_team_list[i].surname);

    }

    // SORTING FUNCTION

    sort(combined_team_list, 0, COMB - 1);

    printf("\n****************************************************************
*********\n");

    printf("\nSorted List of Employees in Surname Order:\n\n");


    //Displays every employee's details
    for (i = 0; i < COMB; i++)
    {
        printf("Name: %s, ", combined_team_list[i].surname);
        printf("%s\n", combined_team_list[i].firstname);
        printf("Team: %c\n", combined_team_list[i].team);
        printf("Line #: %d\n", combined_team_list[i].lineNumber);
        printf("ID: %d\n", combined_team_list[i].cert.employee_ID);
        printf("Number of Certs: %d\n",
combined_team_list[i].cert.cert_line_ID);
        printf("-------------------\n");
    }

    printf("\n****************************************************************
*********\n");

    // Certified list function
    fully_certified();

    printf("\n****************************************************************
*********\n");

    // Search by surname function
    char surname_search[NAME];

    printf("\nEnter the surname of the person that you wish to search
up:\n\n");
    scanf("%s", &surname_search);

    int search_result = search(surname_search); // function called here

    if (search_result == -1)
    {
        printf("No Employee found\n");
    }
    else
    {
        printf("\nThis Employee's ID is %d\n", search_result);
    }

    return 0;

} // end of main

// This function divides the elements in the team combined list and sorts it
by calling the merge function below
void sort(struct team_list array[], int low, int high)
{
    if(low < high)
    {
        int mid = (low + high) / 2;
        // first half is sorted, then the second half
        sort(array, low, mid); //calls itself
        sort(array, mid + 1, high);

        merge(array, low, mid, high); // calls merge functions
    }

} // end of function

/*This function is called by the sort function above. It's used to merge the 
two halves.
  To sort in alphabetical order, it must compare strings*/
void merge(struct team_list array[], int low, int mid, int high)
{
    int i, j, k;
    int left = mid - low + 1;
    int right = high - mid;

    //temporary stucture arrays
    struct team_list Left[left];
    struct team_list Right[right];

    // team_combined_list was passed down, and its data is copied into 
temporary structure arrays
    for(i = 0; i < left; i++)
    {
        Left[i] = array[low + i];
    }
    for(j = 0; j < right; j++)
    {
        Right[j] = array[mid + 1 + j];
    }

    // merges the temp structure arrays back into the team_combined_list array
    i = 0;
    j = 0;
    k = low;

    while(i < left && j < right)
    {
        if(strcmp(Left[i].surname, Right[j].surname) < 0)
        {
            array[k] = Left[i];
            i++;
        }
        else
        {
            array[k] = Right[j];
            j++;
        }
        k++;
    }

    //copies the rest of the elements from the temporary arrays into the
combined list array
    while(i < left)
    {
        array[k] = Left[i];
        i++;
        k++;
    }

    while(i < right)
    {
        array[k] = Left[j];
        j++;
        k++;
    }

} // end of function

// This function is used to produce the list of workers that are fully
certified
void fully_certified()
{
    int i;

    for (i = 0; i < COMB ; i++)
    {
        if(combined_team_list[i].cert.cert_line_ID == 3)
        {
            printf("\n%s %s has certifications in all 3 lines\n",
combined_team_list[i].firstname, combined_team_list[i].surname);
        }
    }

} // end of function

// This function is used to allow user to search for an employee's ID by
entering their surname. It does this by comparing strings
int search(char lastname[NAME])
{
    int i;

    for (i = 0; i < COMB; i++)
    {
        if (strcmp(combined_team_list[i].surname, lastname) == 0)
        {
            return combined_team_list[i].cert.employee_ID;
        }
    }

} // end of function
```