פרויקט גמר
מעבדה מתקדמת בתכנות

פבל גייץ 324392729

TWO TYPES OF STRUCTS FOR TWO LINKED LISTS (ITEMS & USERS)

```c
struct system_user
{
    char username[16];
    char password[16];
    char fullname[21];
    int lvl;
    int id;
    struct system_user* next;
    struct system_user* prev;
};

struct item
{
    int id;
    int day;
    int month;
    int year;
    int hours;
    int mins;
    char product_type[20];
    char product_name[20];
    char date[18];
    float price;
    char in_store[12];
    struct item* next;
    struct item* prev;
};
```

SYSTEM ENTRY FUNCTION

```c
int system_enter(char *path, struct system_user ** s_user)
{
    int day, month, year, hours, mins;
    int counter = 0, level;
    char username_enter[16], password_enter[16];
    char temp[100], username[16], password[16], fullname[21];
    struct system_user* tmp_user = (struct system_user*)malloc(sizeof(struct system_user));
    while(counter<3)
    {
        printf("Enter your Username: \n");
        scanf("%[^\n]%*c", username_enter);
        printf("Enter your Password: \n");
        scanf("%[^\n]%*c", password_enter);

        FILE *fp = fopen(path, "r");
        if (!fp)
        {
            create_users(USERS_PATH);
            printf("File 'users.txt' was created. Enter once again please!\n");
            //LOG INFO
            FILE *log = fopen(LOG_PATH, "a");
            getDateTime(&day, &month, &year, &hours, &mins);
            fprintf(log,"%02d/%02d/%d, %02d:%02d : File 'users.txt' was created.\n\n", day, month, year, hours, mins);
            fclose(log);
        }
        else
        {
            fgets(temp, 100, fp);
            while (fscanf(fp, "%15s %15s %15d %20s", username, password, &level, fullname) != EOF)
            {
                if(strcmp(username_enter, username) == 0 && strcmp(password_enter, password) == 0)
                {
                    counter = 3;
                    strcpy(tmp_user->username, username);
                    strcpy(tmp_user->password, password);
                    strcpy(tmp_user->fullname, fullname);
                    tmp_user->lvl = level;
                    *s_user = tmp_user;
                    //LOG INFO
                    FILE *log = fopen(LOG_PATH, "a");
                    getDateTime(&day, &month, &year, &hours, &mins);
                    fprintf(log,"%02d/%02d/%d, %02d:%02d : User %s entered the system.\n", day, month, year, hours, mins, tmp_user->fullname);
                    fclose(log);
                    return 1;
                }
            }
            printf("Username is incorrect!\n");
            counter++;
        }
        fclose(fp);
    }
    //LOG INFO
    FILE *log = fopen(LOG_PATH, "a");
    getDateTime(&day, &month, &year, &hours, &mins);
    fprintf(log,"%02d/%02d/%d, %02d:%02d : Login fail.\n", day, month, year, hours, mins);
    fclose(log);
    return 0;
}
```

If the «users.txt» doesn't exist, we create a new file with system manager account.

```c
void create_users(char *path)
{
    FILE *fp = fopen(path, "w");
    if (fp == NULL)
    {
        printf("Unable to create file!\n");
        exit(2);
    }
    fputs("Username        Password        L Fullname             ", fp);  fputs("\n", fp);
    fputs("admin           admin           3 System_Manager       ", fp);  fputs("\n", fp);
    fclose(fp);
}
```

READING «ITEMS.TXT» AND CREATING LINKED LIST OF ITEMS

```c
struct item* readItems(char *path, struct item* head, struct system_user* main_user, int print)
{
    int day, month, year, hours, mins;
    int id;
    float price;
    char product_type[20], product_name[20], in_store[12], temp[82];
    struct item* tmp_head = NULL;
    FILE *fp = fopen(path, "r");
    while (!fp)
    {
        fclose(fp);
        create_items(ITEMS_PATH);
        printf("File 'items.txt' was created");
        //LOG INFO
        FILE *log = fopen(LOG_PATH, "a");
        getDateTime(&day, &month, &year, &hours, &mins);
        fprintf(log,"%02d/%02d/%d, %02d:%02d : User %s created 'items.txt'.\n\n", day, month, year, hours, mins, main_user->fullname);
        fclose(log);
        return tmp_head;
    }
    fgets(temp, 85, fp);
    if(print == 1){
        puts(temp);
    }

    ...........

    ...........
```

If the «items.txt» doesn't exist, we create a new empty file.

```c
void create_items(char *path)
{
    FILE *fp = fopen(path, "w");
    if (fp == NULL)
    {
        printf("Unable to create file!\n");
        exit(2);
    }
    fputs("ID      Product type        Product name       Entry Date        In store     Price    ", fp);  fputs("\n", fp);
    fclose(fp);
}
```

```c
    ...........
    ...........

while (fscanf(fp, "%d %20s %20s %2d/%2d/%d, %02d:%02d %12s %f\n", &id, product_type, product_name, &day, &month, &year, &hours, &mins,
in_store, &price) != EOF)
    {
        struct item* tmp = (struct item*)malloc(sizeof(struct item));

        struct item* tmp_head2 = head;
        if(print == 1){
            printf("%-7d %-19s %-18s %-2d/%-d/%-4d,%-2d:%-2d   %-11s %-10.2f\n", id, product_type, product_name, day, month, year, hours,
mins, in_store, price);
        }
        tmp->id = id;
        strcpy(tmp->product_type, product_type);
        strcpy(tmp->product_name, product_name);
        strcpy(tmp->date, "date");
        strcpy(tmp->in_store, in_store);
        tmp->day = day;
        tmp->month = month;
        tmp->year = year;
        tmp->hours = hours;
        tmp->mins = mins;
        tmp->price = price;
        tmp->next = NULL;
        if(tmp_head == NULL){
            tmp_head = tmp;
            tmp_head->prev = NULL;
        }
        else{
            while(tmp_head->next != NULL){
                tmp_head = tmp_head->next;
            }
            tmp_head2 = tmp_head;
            tmp_head->next = tmp;
            tmp_head->next->prev = tmp_head2;
            while(tmp_head->prev != NULL){
                tmp_head = tmp_head->prev;
            }
        }
    }
    fclose(fp);
    //LOG INFO
    FILE *log = fopen(LOG_PATH, "a");
    getDateTime(&day, &month, &year, &hours, &mins);
    fprintf(log,"%02d/%02d/%d, %02d:%02d : User %s printed the list of items.\n", day, month, year, hours, mins, main_user->fullname);
    fclose(log);
    return tmp_head;
}
```

READING «USERS.TXT» AND CREATING LINKED LIST OF USERS

```c
struct system_user* read_users(char *path, struct system_user* head_of_users, struct system_user* main_user)
{
    int day, month, year, hours, mins;
    int level, id = 1;
    char temp[55], username[16], password[16], fullname[21];
    struct system_user* tmp_head = NULL;

    FILE *fp = fopen(path, "r");
    while (!fp)
    {
        printf("File not found!\n");
    }
    fgets(temp, 55, fp);
    puts(temp);
    while (fscanf(fp, "%15s %15s %d %20s", username, password, &level, fullname) != EOF)
    {
        struct system_user* tmp = (struct system_user*)malloc(sizeof(struct system_user));
        struct system_user* tmp_head2 = head_of_users;

        printf("%-15s %-15s %-1d %-19s \n", username, password, level, fullname);

        tmp->id = id;
        strcpy(tmp->username, username);
        strcpy(tmp->password, password);
        tmp->lvl = level;
        strcpy(tmp->fullname, fullname);
        tmp->next = NULL;
        id++;
        if(tmp_head == NULL){
            tmp_head = tmp;
            tmp_head->prev = NULL;
        }
        else{
            while(tmp_head->next != NULL){
                tmp_head = tmp_head->next;
            }
            tmp_head2 = tmp_head;
            tmp_head->next = tmp;
            tmp_head->next->prev = tmp_head2;
            while(tmp_head->prev != NULL){
                tmp_head = tmp_head->prev;
            }
        }
    }
    fclose(fp);
    //LOG INFO
    FILE *log = fopen(LOG_PATH, "a");
    getDateTime(&day, &month, &year, &hours, &mins);
    fprintf(log,"%02d/%02d/%d, %02d:%02d : User %s printed the list of users.\n", day, month, year, hours, mins, main_user->fullname);
    fclose(log);
    return tmp_head;
}
```

ADDING NEW ITEM TO THE LINKED LIST OF ITEMS AND TXT FILE

```c
struct item * add_new_item(char *path, struct item* head, struct system_user* main_user)
{
    int day, month, year, hours, mins;
    float price;
    char product_type[20], product_name[20], date[18] = "date", in_store[12];
    struct item* tmp = (struct item*)malloc(sizeof(struct item));
    struct item* tmp_head = (struct item*)malloc(sizeof(struct item));
    tmp_head = head;
    printf("Enter product type: ");
    scanf(" %[^\n]%*c", product_type);
    printf("Enter product name: ");
    scanf(" %[^\n]%*c", product_name);
    printf("Product in store? ");
    scanf(" %[^\n]%*c", in_store);
    printf("Enter product price:");
    scanf(" %f",&price);

    FILE *fp = fopen(path, "a");

    if(tmp_head == NULL){
        tmp->id = 1;
        strcpy(tmp->product_type, product_type);
        strcpy(tmp->product_name, product_name);
        strcpy(tmp->date, date);
        strcpy(tmp->in_store, in_store);
        tmp->price = price;
        tmp->next = NULL;
        tmp->prev = NULL;
        tmp_head = tmp;
        getDateTime(&day, &month, &year, &hours, &mins);
        tmp->day = day;
        tmp->month = month;
        tmp->year = year;
        tmp->hours = hours;
        tmp->mins = mins;
        fprintf(fp,"%-7d %-19s %-18s %02d/%02d/%d,%02d:%02d  %-11s %-7.2f \n", tmp->id, product_type, product_name, day, month, year, hours,
mins, in_store, price);
        //LOG INFO
        FILE *log = fopen(LOG_PATH, "a");
        getDateTime(&day, &month, &year, &hours, &mins);
        fprintf(log,"%02d/%02d/%d, %02d:%02d : User %s added new item - %s / %s / %.2f.\n", day, month, year, hours, mins, main_user-
>fullname, product_type, product_name, price);
        fclose(log);
    }
..
..
..
..
..
..
    else{
        while(tmp_head->next != NULL){
            tmp_head = tmp_head->next;
        }
        tmp->id = tmp_head->id+1;
        strcpy(tmp->product_type, product_type);
        strcpy(tmp->product_name, product_name);
        strcpy(tmp->date, date);
        strcpy(tmp->in_store, in_store);
        tmp->price = price;
        tmp->next = NULL;
        tmp->prev = tmp_head;
        tmp_head->next = tmp;
        getDateTime(&day, &month, &year, &hours, &mins);
        tmp->day = day;
        tmp->month = month;
        tmp->year = year;
        tmp->hours = hours;
        tmp->mins = mins;
        fprintf(fp,"%-7d %-19s %-18s %02d/%02d/%d,%02d:%02d  %-11s %-7.2f \n", tmp->id, product_type, product_name, day, month, year, hours,
mins, in_store, price);
        //LOG INFO
        FILE *log = fopen(LOG_PATH, "a");
        getDateTime(&day, &month, &year, &hours, &mins);
        fprintf(log,"%02d/%02d/%d, %02d:%02d : User %s added new item - %s / %s / %.2f.\n", day, month, year, hours, mins, main_user-
>fullname, product_type, product_name, price);
        fclose(log);
        while(tmp_head->prev != NULL){
            tmp_head = tmp_head->prev;
        }
    }
    fclose(fp);
    return tmp_head;

}
```

ADDING NEW USER TO THE LINKED LIST OF USERS AND TXT FILE

```c
struct system_user* add_new_user(char *path, struct system_user* head_of_users, struct system_user* main_user)
{
    int day, month, year, hours, mins;
    int level;
    char username[16], password[16], fullname[21];
    struct system_user* tmp = (struct system_user*)malloc(sizeof(struct system_user));
    struct system_user* tmp_head = (struct system_user*)malloc(sizeof(struct system_user));
    tmp_head = head_of_users;
    printf("Enter user name: ");
    scanf(" %[^\n]%*c", username);
    printf("Enter password: ");
    scanf(" %[^\n]%*c", password);
    printf("Enter full name: ");
    scanf(" %[^\n]%*c", fullname);
    printf("Enter access level:");
    scanf(" %d",&level);
    FILE *fp = fopen(path, "a");

    while(tmp_head->next != NULL){
        tmp_head = tmp_head->next;
    }
    tmp->id = tmp_head->id+1;
    strcpy(tmp->username, username);
    strcpy(tmp->password, password);
    strcpy(tmp->fullname, fullname);
    tmp->lvl = level;

    tmp->next = NULL;
    tmp->prev = tmp_head;
    tmp_head->next = tmp;
    fprintf(fp,"%-15s %-15s %-1d %-19s \n", username, password, level, fullname);
    //LOG INFO
    FILE *log = fopen(LOG_PATH, "a");
    getDateTime(&day, &month, &year, &hours, &mins);
    fprintf(log,"%02d/%02d/%d, %02d:%02d : User %s added new user - %s with access level %d.\n", day, month, year, hours, mins, main_user-
>fullname, fullname, level);
    fclose(log);
    while(tmp_head->prev != NULL){
        tmp_head = tmp_head->prev;
    }
    fclose(fp);
    return tmp_head;
}
```

DELETING ITEM FROM THE LINKED LIST AND RECREATING THE TXT FILE

```c
struct item* delete_item(char *path, struct item* head, struct system_user* main_user)
{
    int day, month, year, hours, mins;
    char product_type[20] = "Product type", product_name[20] = "Product name", date[18] = "Entry Date", in_store[12] = "In store", id[8] =
"ID", price[7] = "Price";
    int i;
    head = readItems(path, head, main_user, 0);
    printf("What item to delete? Choose ID: \n");
    scanf("%d", &i);
    //Deleting for Linked List
    struct item* tmp = (struct item*)malloc(sizeof(struct item));
    struct item* tmp_head = (struct item*)malloc(sizeof(struct item));
    tmp = head;
    tmp_head = head;

    while(tmp->id != i){
        tmp = tmp->next;
        if(tmp == NULL){
            printf("There is no such ID");
            return tmp_head;
        }
    }
    //LOG INFO
    FILE *log = fopen(LOG_PATH, "a");
    getDateTime(&day, &month, &year, &hours, &mins);
    fprintf(log,"%02d/%02d/%d, %02d:%02d : User %s deleted item - %s / %s / %.2f.\n", day, month, year, hours, mins, main_user->fullname,
tmp->product_type, tmp->product_name, tmp->price);
    fclose(log);

    if(tmp->prev == NULL){
        tmp_head = tmp->next;
        tmp_head->prev = NULL;
        return tmp_head;
    }
    tmp_head = tmp->prev;
    tmp_head->next = tmp->next;
    tmp_head->prev = tmp->prev->prev;
    while(tmp_head->prev != NULL){
        tmp_head = tmp_head->prev;
    }
    //Sorting IDS
    tmp = tmp_head;
    int k = 1;
    while(tmp != NULL){
        tmp->id = k;
        k++;
        tmp = tmp->next;
    }
    ..
    ..
    ..
    ..

    //Second part of correcting the file
    tmp = tmp_head;
    FILE *fp = fopen(path, "w");
    if (fp == NULL)
    {
        printf("Unable to create file!\n");
        exit(2);
    }
    fputs(id, fp); fputs("      ", fp); fputs(product_type, fp); fputs("        ", fp); fputs(product_name, fp); fputs("       ",
fp);fputs(date, fp);fputs("        ", fp); fputs(in_store, fp); fputs("    ", fp);fputs(price, fp); fputs("\n", fp);
    getDateTime(&day, &month, &year, &hours, &mins);
    while(tmp !=  NULL){
        fprintf(fp,"%-7d %-19s %-18s %-2d/%-d/%-4d,%-2d:%-2d  %-11s %-10.2f\n", tmp->id, tmp->product_type, tmp->product_name, tmp->day,
tmp->month, tmp->year, tmp->hours, tmp->mins, tmp->in_store, tmp->price);
        tmp = tmp->next;
    }
    fclose(fp);
    free(tmp);
    return tmp_head;
}
```

DELETING USER FROM THE LINKED LIST AND RECREATING THE TXT FILE

```c
struct system_user* delete_user(char *path, struct system_user* head_of_users, struct system_user* main_user)
{
    int day, month, year, hours, mins;
    char username[16] = "username", password[16] = "password", level[2] = "L", fullname[21] = "fullname";
    head_of_users = read_users(path, head_of_users, main_user);
    struct system_user* tmp = (struct system_user*)malloc(sizeof(struct system_user));
    struct system_user* tmp_head = (struct system_user*)malloc(sizeof(struct system_user));
    tmp = head_of_users;
    tmp_head = head_of_users;

    while(tmp != NULL){
        printf("User %d: %s, level of access is %d \n", tmp->id, tmp->fullname, tmp->lvl);
        tmp = tmp->next;
    }
    int user_num;
    printf("Enter user number: ");
    scanf(" %d", &user_num);
    while(user_num == 1){
        printf("You cant delete 'admin' user \n");
        printf("Choose another user: ");
        scanf(" %d", &user_num);
    }
    tmp = head_of_users;
    while(tmp->id != user_num){
        tmp = tmp->next;
        if(tmp == NULL){
        printf("There is no such ID");
        return tmp_head;
        }
    }
    //LOG INFO
    FILE *log = fopen(LOG_PATH, "a");
    getDateTime(&day, &month, &year, &hours, &mins);
    fprintf(log,"%02d/%02d/%d, %02d:%02d : User %s deleted user %s.\n", day, month, year, hours, mins, main_user->fullname, tmp->fullname);
    fclose(log);

    if(tmp->prev == NULL){
        tmp_head = tmp->next;
        tmp_head->prev = NULL;
        return tmp_head;
    }
    tmp_head = tmp->prev;
    tmp_head->next = tmp->next;
    tmp_head->prev = tmp->prev->prev;
    while(tmp_head->prev != NULL){
        tmp_head = tmp_head->prev;
    }
    ..
    ..
    ..
    ..
    //Sorting IDS
    tmp = tmp_head;
    int k = 1;
    while(tmp != NULL){
        tmp->id = k;
        k++;
        tmp = tmp->next;
    }
    //Second part of correcting the file
    tmp = tmp_head;
    FILE *fp = fopen(path, "w");
    if (fp == NULL)
    {
        printf("Unable to create file!\n");
        exit(2);
    }
    fputs(username, fp); fputs("        ", fp); fputs(password, fp); fputs("        ", fp); fputs(level, fp); fputs(" ", fp);fputs(fullname,
fp);fputs("             ", fp); fputs("\n", fp);
    while(tmp !=  NULL){
        fprintf(fp,"%-15s %-15s %-1d %-19s \n", tmp->username, tmp->password, tmp->lvl, tmp->fullname);
        tmp = tmp->next;
    }
    fclose(fp);
    free(tmp);


    return tmp_head;
}
```

FREE MEMORY OF LINKED LISTS

```
void free_items(struct item* head)
{
   struct item* tmp;
   while (head != NULL)
    {
       tmp = head;
       head = head->next;
       free(tmp);
    }
}

void free_system_user(struct system_user* head)
{
   struct system_user* tmp;
   while (head != NULL)
    {
       tmp = head;
       head = head->next;
       free(tmp);
    }
}
```

MAIN

```c
int main()
{
    int day, month, year, hours, mins;
    int access, lvl;
    int i = 100;
    struct system_user* main_user = (struct system_user*)malloc(sizeof(struct system_user));
    struct item* head = NULL;
    struct system_user* head_of_users = NULL;
    //User entry to the system
    access = system_enter(USERS_PATH, &main_user);
    if(access == 0)
    {
        printf("You are blocked forever!\n");
        exit(1);
    }
    printf("Welcome back, %s! Choose your actions:\n", main_user->fullname);
    lvl = main_user->lvl;

    while(i != 0){

            if(lvl == 1)
            {
                printf("1. List of items // 2. Add item // ");
                printf("3. Sort by product type or name // 4. Sort by availability in store // 0. EXIT \n");
                scanf("%d", &i);
                switch (i) {
                    case 1:
                        head = readItems(ITEMS_PATH, head, main_user, 1);
                        break;

                    case 2:
                        head = readItems(ITEMS_PATH, head, main_user, 0);
                        head = add_new_item(ITEMS_PATH, head, main_user);
                        break;
                    case 3:
                        head = sort_by_product_type(ITEMS_PATH, head, main_user);
                        break;
                    case 4:
                        head = sort_by_in_store(ITEMS_PATH, head, main_user);
                        break;
                }
            }
            ..
            ..
```

```
..
..

if(lvl == 2)
        {
            printf("1. List of items // 2. Add item // 3. Update item // 4. Delete item \n");
            printf("5. Sort by product type or name // 6. Sort by availability in store // 0. EXIT\n");
            scanf("%d", &i);
            switch (i) {
                case 1:
                    head = readItems(ITEMS_PATH, head, main_user, 1);
                    break;

                case 2:
                    head = readItems(ITEMS_PATH, head, main_user, 0);
                    head = add_new_item(ITEMS_PATH, head, main_user);
                    break;
                case 3:
                    head = update_item(ITEMS_PATH, head, main_user);
                    break;
                case 4:
                    head = delete_item(ITEMS_PATH, head, main_user);
                    break;
                case 5:
                    head = sort_by_product_type(ITEMS_PATH, head, main_user);
                    break;
                case 6:
                    head = sort_by_in_store(ITEMS_PATH, head, main_user);
                    break;
            }
        }
..
..
```

```
..
..

if(lvl == 3)
        {
            printf("1. List of items // 2. Add item // 3. Update item // 4. Delete item \n");
            printf("5. List of users // 6. Add user // 7. Update user // 8. Delete user \n");
            printf("9. Sort by product type or name // 10. Sort by availability in store // 0. EXIT\n");
            scanf("%d", &i);
            switch (i) {
                case 1:
                    head = readItems(ITEMS_PATH, head, main_user, 1);
                    break;

                case 2:
                    head = readItems(ITEMS_PATH, head, main_user, 0);
                    head = add_new_item(ITEMS_PATH, head, main_user);
                    break;
                case 3:
                    head = update_item(ITEMS_PATH, head, main_user);
                    break;
                case 4:
                    head = delete_item(ITEMS_PATH, head, main_user);
                    break;
                case 5:
                    head_of_users = read_users(USERS_PATH, head_of_users, main_user);
                    break;
                case 6:
                    head_of_users = read_users(USERS_PATH, head_of_users, main_user);
                    head_of_users = add_new_user(USERS_PATH, head_of_users, main_user);
                    break;
                case 7:
                    head_of_users = update_user(USERS_PATH, head_of_users, main_user);
                    break;
                case 8:
                    head_of_users = delete_user(USERS_PATH, head_of_users, main_user);
                    break;
                case 9:
                    head = sort_by_product_type(ITEMS_PATH, head, main_user);
                    break;
                case 10:
                    head = sort_by_in_store(ITEMS_PATH, head, main_user);
                    break;
            }
        }
        printf("\n");
    }
..
..
```

```
..
..
//LOG INFO
    FILE *log = fopen(LOG_PATH, "a");
    getDateTime(&day, &month, &year, &hours, &mins);
    fprintf(log,"%02d/%02d/%d, %02d:%02d : User %s loged out from the system.\n\n", day, month, year, hours, mins, main_user->fullname);
    fclose(log);

    //FREE MEMORY
    free(main_user);
    free_items(head);
    free_system_user(head_of_users);
    return 0;
}
```