



Технически университет – София

Катедра „Компютърни системи“

Курсова работа

ПО

**Програмиране и Използване на
Компютри II**

Студент: Николай Георгиев Станишев

Факултет: ФКСТ

Група: 52

Факултетен №: 121218038

Задание № 8

Ръководител: ас. маг. инж. Момчил Петков

Дата:

София
2019

1. Задание № 8

Да се разработи програма тип „меню” за поддържане на складово стопанство със следните изисквания:

1.1 Наличностите в складовете се съхраняват в двоични файлове. За всяка стока в склада се пазят следните данни:

- 1) Номенклатурен номер - уникално 12 цифрено число;
- 2) Наименование на стоката - до 50 символен низ;
- 3) Единична цена - реално число;
- 4) Количество - цяло число;
- 5) Дата на производство на стоката - записана във формата ДД.ММ.ГГГГ.

1.2 Да се извършват следните обработки:

- 1) добавя нова стока в склада.
- 2) променяне на наличностите от зададена стока (добавяне или изваждане количества от склада). При изваждане ако заявката е за повече съществуващото количество - да не се изпълнява. Ако наличностите станат равни на 0 стоката да се изключи от склада.
- 3) извеждане информация за всички стоки от склада с изтекъл срок на годност (текущата дата се задава от потребителя).
- 4) По зададен идентификационен номер да се разпечатва информацията за доставена стока.

1.3 Данните да се поддържат в динамична структура - едносвързан списък в оперативната памет на ПК.

1.4 ИЗИСКВАНИЯ КЪМ ОФОРМЛЕНИЕТО

Задачата да се оформи като задача, съдържаща:

- 1) титулна страница с данни за студента, ръководителя на курсовата задача;
- 2) текст на заданието;
- 3) описание на използваните модули (функции) - прототип, входно изходни параметри и предназначение;
- 4) общо описание за функциониране на програмата (вход/изход);
- 5) листинг на source (изходния) код на програмата;
- 6) резултати от изпълнението на програмата (контролен пример);

2 Описание на използваните модули(функции) – прототип, входно изходни параметри и предназначение

2.1 load_products

```
int load_products(char *filename, struct products *products);
```

1. Входни параметри
 - 1.1. filename – Име на входният файл, в който се съдържа базата данни
 - 1.2. products – Заредената база данни от продукти в паметта
2. Изходна стойност
 - 2.1. error_code – Код за грешка. Приемани стойности:
 - 2.1.1. 0 – Всичко е изпълнено без проблем
 - 2.1.2. -1 – Има настъпила грешка
3. Функционалност – Функция за зареждане на стоките в паметта от зададен входен файл

2.2 save_products

```
int save_products(char* filename, struct products *products);
```

1. Входни параметри
 - 1.1. filename – Име на изходния файл, в който се съдържа базата данни
 - 1.2. products – Заредената база данни от продукти в паметта
2. Изходна стойност
 - 2.1. error_code – Код за грешка. Приемани стойности:
 - 2.1.1. 0 – Всичко е изпълнено без проблем
 - 2.1.2. -1 – Има настъпила грешка
3. Функционалност – Функция за запазване на стоките в базата данни в зададен изходен файл

2.3 find_by_id

```
struct products* find_by_id(struct products *products, long id);
```

1. Входни параметри
 - 1.1. products – Заредената база данни от продукти в паметта
 - 1.2. id – Номенклатурният номер на търсеният продукт
2. Изходна стойност
 - 2.1. products_with_id – Елементът от структурата products с номенклатурен номер id
3. Функционалност – Функция за вземане на елемент от списъка products по зададен номенклатурен номер.

2.4 delete_by_id

```
void delete_by_id(struct products *products, long id);
```

1. Входни параметри

- 1.1. products – Заредената база данни от продукти в паметта
- 1.2. id – Номенклатурният номер на търсеният продукт за изтриване
- 2. Функционалност – Функция за изтриване на елемент от списъка products по зададен номенклатурен номер.

2.5 string_to_time

```
time_t string_to_time(char *string_time);
```

- 1. Входни параметри
 - 1.1. string_time – Времето записано в символен низ
- 2. Изходна стойност
 - 2.1. time_t_time – Времето записано в time_t структура
- 3. Функционалност – Функция за превръщане на време в формат на символен низ в такова във формат на time_t структура

2.6 add_new_product

```
int add_new_product(struct products *products);
```

- 1. Входни параметри
 - 1.1. products – Заредената база данни от продукти в паметта
- 2. Изходна стойност
 - 2.1. error_code – Код за грешка. Приемани стойности:
 - 2.1.1. 0 – Всичко е изпълнено без проблем
 - 2.1.2. -1 – Има настъпила грешка
- 3. Функционалност – Функция за добавяне на нова стока в свързаният списък списък репрезентиращ склада

2.7 print_product_with_id

```
int print_product_with_id(struct products *products);
```

- 1. Входни параметри
 - 1.1. products – Заредената база данни от продукти в паметта
- 2. Изходна стойност
 - 2.1. error_code – Код за грешка. Приемани стойности:
 - 2.1.1. 0 – Всичко е изпълнено без проблем
 - 2.1.2. -1 – Има настъпила грешка
- 3. Функционалност – Функция за намиране на стока със зададен номенклатурен номер и разпечатване на неговата информация

2.8 print_all_expired_products

```
int print_all_expired_products(struct products *products);
```

- 1. Входни параметри
 - 1.1. products – Заредената база данни от продукти в паметта

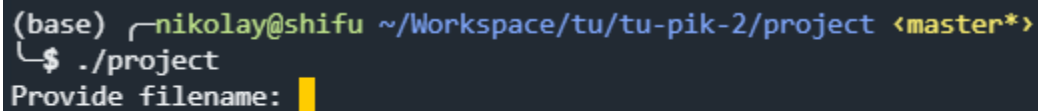
2. Изходна стойност
 - 2.1. error_code – Код за грешка. Приемани стойности:
 - 2.1.1. 0 – Всичко е изпълнено без проблем
 - 2.1.2. -1 – Има настъпила грешка
3. Функционалност – Функция за намирането на стоките с изтекъл срок на годност спрямо зададена от потребителя дата.

2.9 change_quantity_of_product

```
int change_quantity_of_product(struct products *products);
```

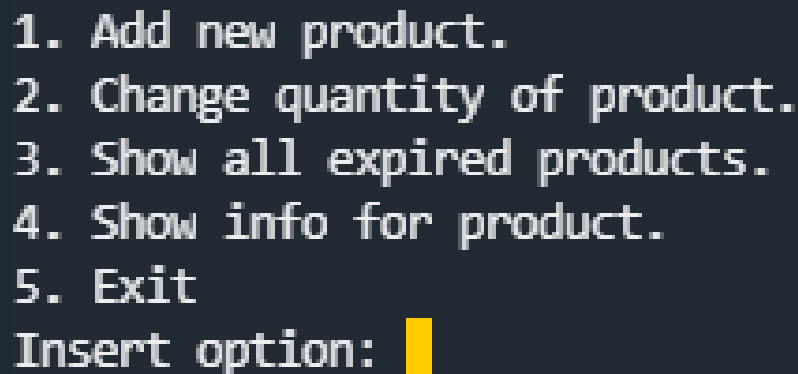
1. Входни параметри
 - 1.1. products – Заредената база данни от продукти в паметта
2. Изходна стойност
 - 2.1. error_code – Код за грешка. Приемани стойности:
 - 2.1.1. 0 – Всичко е изпълнено без проблем
 - 2.1.2. -1 – Има настъпила грешка
3. Функционалност – Функция за промяната на наличността на зададена стойност

3 Общо описание за функциониране на програмата (вход/изход)



```
(base) └─nikolay@shifu ~/Workspace/tu/tu-pik-2/project <master*>
└─$ ./project
Provide filename: █
```

Figure 1



```
1. Add new product.
2. Change quantity of product.
3. Show all expired products.
4. Show info for product.
5. Exit
Insert option: █
```

Figure 2

Програмата представлява конзолно приложение. При стартиране на програмата се зарежда база данни от зададен файл (Figure 1), а при приключването и се запазват данните в зададения файл. Основният екран на програмата е меню с различни опции (Figure 2). Тези опции са:

1. Опция за добавяне на нова стока в склада.
2. Опция за променяне на наличностите от зададена стока.
3. Опция за извеждане информация за всички стоки от склада с изтекъл срок на годност при зададена дата от потребителя.
4. Опция за извеждане на информация за доставена стока по зададен от потребителя идентификационен номер.
5. Опция за изход от програмата. Другата възможност за изход от програмата е EOF.

След задаване на опцията се изпълнява зададената задача към програмата. Менюто се показва до приключване на изпълнението на програмата.

4 Изходен код на програмата

```
#define _XOPEN_SOURCE

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

struct product {
    long id;
    char name[50];
    float price;
    int quantity;
    char date[11];
};

struct products {
    struct product val;

    struct products *next;
};

int load_products(char*, struct products*);
int save_products(char*, struct products*);
struct products* find_by_id(struct products*, long);
void delete_by_id(struct products*, long);
time_t string_to_time(char*);
int add_new_product(struct products*);
int change_quantity_of_product(struct products*);
int print_all_expired_products(struct products*);
int print_product_with_id(struct products*);

int main() {
    char filename[100];
    printf("Provide filename: ");
    scanf("%s", filename);

    struct products *products = NULL;
    products = malloc(sizeof(struct products));
    if (products == NULL) {
        perror("malloc");
        return -1;
    }
}
```

```

products->next = NULL;
if (load_products(filename, products) == -1) {
    return -1;
}

int option = 0;
int exit_flag = 1;
while(exit_flag) {
    system("@cls||clear");

    printf("1. Add new product.\n");
    printf("2. Change quantity of product.\n");
    printf("3. Show all expired products.\n");
    printf("4. Show info for product.\n");
    printf("5. Exit\n");

    printf("Insert option: ");
    if (scanf("%d", &option) == EOF) {
        break;
    }

    switch (option) {
        case 1:
            if (add_new_product(products)) {
                return -1;
            }
            break;
        case 2:
            if (change_quantity_of_product(products)) {
                return -1;
            }
            break;
        case 3:
            if (print_all_expired_products(products)) {
                return -1;
            }
            break;
        case 4:
            if (print_product_with_id(products)) {
                return -1;
            }
            break;
        case 5:
            exit_flag = 0;
            break;
    }
}

```



```

        default:
            break;
    }

    printf("Continue");
    char ch;
    scanf("%c",&ch);
    getchar();
}

if (save_products(filename, products)) {
    return -1;
}

return 0;
}

int load_products(char *filename, struct products *products) {
    FILE *fp = NULL;

    if ((fp = fopen(filename, "rb")) == NULL) {
        perror("fopen");
        return -1;
    }

    struct products *prev_products = products;
    struct products *curr_products = NULL;
    while (1) {
        struct product product;

        if (fread(&product, sizeof(struct product), 1, fp) != 1) {
            if (feof(fp)) {
                break;
            }
            fclose(fp);
            perror("fread");
            return -1;
        }

        curr_products = malloc(sizeof(struct products));
        if (curr_products == NULL) {
            perror("malloc");
            return -1;
        }
        curr_products->val = product;
    }
}

```

```

    curr_products->next = NULL;

    prev_products->next = curr_products;
    prev_products = curr_products;
}

fclose(fp);

return 0;
}

int save_products(char* filename, struct products *products) {
    FILE *fp = NULL;

    if ((fp = fopen(filename, "wb")) == NULL) {
        perror("fopen");
        return -1;
    }

    while (1) {
        products = products->next;
        if (products == NULL) {
            break;
        }

        if (fwrite(&products->val, sizeof(struct product), 1, fp) != 1) {
            fclose(fp);
            perror("fwrite");
            return -1;
        }
    }

    return 0;
}

struct products* find_by_id(struct products *products, long id) {
    struct products *iterate_products = products->next;
    while(1) {
        if (iterate_products == NULL) {
            break;
        }

        if (iterate_products->val.id == id) {
            return iterate_products;
        }
    }
}

```

```

        iterate_products = iterate_products->next;
    }

    return NULL;
}

void delete_by_id(struct products *products, long id) {
    struct products *prev_products = products;
    struct products *iterate_products = products->next;
    while(1) {
        if (iterate_products == NULL) {
            break;
        }

        if (iterate_products->val.id == id) {
            prev_products->next = iterate_products->next;
            break;
        }

        iterate_products = iterate_products->next;
        prev_products = prev_products->next;
    }
}

time_t string_to_time(char *string_time) {
    time_t result = 0;

    int day = 0, month = 0, year = 0;

    if (sscanf(string_time, "%2d.%2d.%4d", &day, &month, &year) == 3) {
        struct tm time = {0};
        time.tm_year = year - 1900;
        time.tm_mon = month - 1;
        time.tm_mday = day;

        if ((result = mktime(&time)) == (time_t) - 1) {
            perror("mktime");
            return -1;
        }
    } else {
        perror("sscanf");
        return -1;
    }
}

```

```

    return result;
}

int add_new_product(struct products *products) {
    long id = 0;
    char name[50] = "";
    float price;
    int quantity;
    char date[11] = "";

    printf("Insert id: ");
    scanf("%ld", &id);
    printf("Insert name: ");
    scanf("%s", name);
    printf("Insert price: ");
    scanf("%f", &price);
    printf("Insert quantity: ");
    scanf("%d", &quantity);
    printf("Insert date /in dd.mm.yyyy format/: ");
    scanf("%s", date);

    struct product product;
    product.id = id;
    strcpy(product.name, name);
    product.price = price;
    product.quantity = quantity;
    strcpy(product.date, date);

    struct products *last_products = products;
    while(last_products->next) {
        last_products = last_products->next;
    }

    struct products *next_products = NULL;
    next_products = malloc(sizeof(struct products));
    if (next_products == NULL) {
        perror("malloc");
        return -1;
    }
    next_products->val = product;
    next_products->next = NULL;

    last_products->next = next_products;

    return 0;
}

```

```

}

int print_product_with_id(struct products *products) {
    long id;
    printf("Insert id: ");
    scanf("%ld", &id);

    struct products *products_with_id = find_by_id(products, id);

    if (products_with_id == NULL) {
        printf("Missing product!\n");
    } else {
        printf("id: %ld\n", products_with_id->val.id);
        printf("name: %s\n", products_with_id->val.name);
        printf("price: %.2f\n", products_with_id->val.price);
        printf("quantity: %d\n", products_with_id->val.quantity);
        printf("date: %s\n", products_with_id->val.date);
    }

    return 0;
}

int print_all_expired_products(struct products *products) {
    char date[11];
    printf("Insert date to compare /in dd.mm.yyyy format/: ");
    scanf("%s", date);

    time_t date_t = string_to_time(date);
    if (date_t == -1) {
        return -1;
    }

    struct products *iterate_products = products->next;
    while(1) {
        if (iterate_products == NULL) {
            break;
        }

        time_t product_date_t = string_to_time(iterate_products->val.date);
        if (product_date_t == -1) {
            return -1;
        }

        if (difftime(product_date_t, date_t) < 0) {
            printf("id: %ld\n", iterate_products->val.id);
        }
    }
}

```

```

        printf("name: %s\n", iterate_products->val.name);
        printf("price: %.2f\n", iterate_products->val.price);
        printf("quantity: %d\n", iterate_products->val.quantity);
        printf("date: %s\n", iterate_products->val.date);
    }

    iterate_products = iterate_products->next;
}

return 0;
}

int change_quantity_of_product(struct products *products) {
    long id;
    printf("Insert id: ");
    scanf("%ld", &id);

    int quantity;
    printf("Insert quantity to change: ");
    scanf("%d", &quantity);

    struct products *products_with_id = find_by_id(products, id);

    if (products_with_id == NULL) {
        printf("Missing product!\n");
    } else {
        int new_quantity = products_with_id->val.quantity + quantity;
        if (new_quantity < 0) {
            printf("Quantity to change is bigger than product quantity!\n");
        } else if (new_quantity > 0) {
            products_with_id->val.quantity = new_quantity;
        } else if (new_quantity == 0) {
            delete_by_id(products, id);
        }
    }

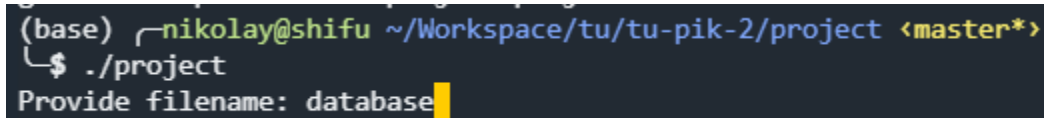
    return 0;
}

```

5 Резултати от изпълнението на програмата (контролен пример)

5.1 Зареждане на базата данни от файл

В началото на изпълнение на програмата се изисква задаване на файл, който представлява базата данни (Figure 3).



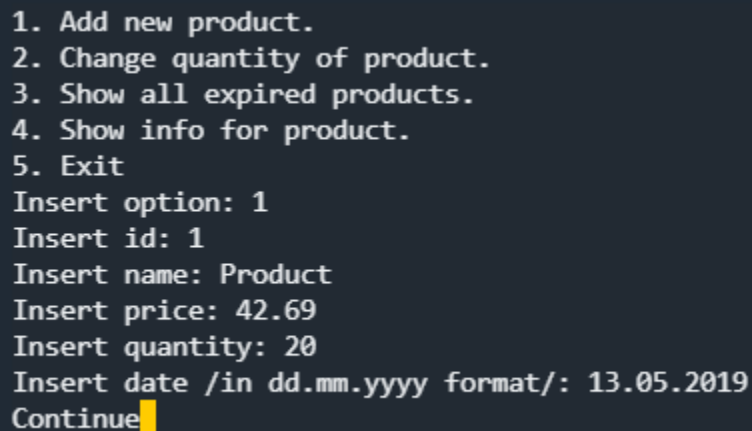
```
(base) └─nikolay@shifu ~/Workspace/tu/tu-pik-2/project <master*>
└─$ ./project
Provide filename: database
```

Figure 3

5.2 Добавяне на нова стока в склада

При опцията добавяне на продукт се добавя стока в склада. Параметрите, които са нужни за добавяне на стока са:

- 1) Номенклатурен номер
- 2) Наименование на стоката
- 3) Единична цена
- 4) Количество
- 5) Дата на производство на стоката



```
1. Add new product.
2. Change quantity of product.
3. Show all expired products.
4. Show info for product.
5. Exit
Insert option: 1
Insert id: 1
Insert name: Product
Insert price: 42.69
Insert quantity: 20
Insert date /in dd.mm.yyyy format/: 13.05.2019
Continue
```

Figure 4

5.3 Промяна на наличността от зададена стока

Опцията за променяне на наличността от зададена стока приема два параметри:

- 1) Номенклатурен номер на стоката, на която ще се извършва промяна на наличността.

- 2) Количеството, с което ще се промени наличността на стоката (положително число за увеличаване на количеството на стоката и отрицателно за намаляване на количеството на стоката).

Тя има няколко под опции:

- 1) Добавяне на количество към стоката в склада (Figure 5).

```
1. Add new product.
2. Change quantity of product.
3. Show all expired products.
4. Show info for product.
5. Exit
Insert option: 2
Insert id: 1
Insert quantity to change: 5
Continue
```

Figure 5

- 2) Премахване на количество от стоката от склада (Figure 6).

```
1. Add new product.
2. Change quantity of product.
3. Show all expired products.
4. Show info for product.
5. Exit
Insert option: 2
Insert id: 1
Insert quantity to change: -5
Continue
```

Figure 6

- 3) Премахване на стока от склада (Figure 7). Тази опция влиза в сила, когато се премахне цялото количество на продукта.


```
1. Add new product.  
2. Change quantity of product.  
3. Show all expired products.  
4. Show info for product.  
5. Exit  
Insert option: 2  
Insert id: 1  
Insert quantity to change: -20  
Continue
```

Figure 7

- 4) Опит за промяна на количеството на стока, която не присъства в склада (Figure 8).

```
1. Add new product.  
2. Change quantity of product.  
3. Show all expired products.  
4. Show info for product.  
5. Exit  
Insert option: 2  
Insert id: 10  
Insert quantity to change: 15  
Missing product!  
Continue
```

Figure 8

5.4 Извеждане на стоките с изтекъл срок на годност

Опцията за извеждане на стоките с изтекъл срок на годност извежда стоките с изтекъл срок на годност спрямо зададена дата (Figure 9). Тя приема два параметъра:

- 1) Номенклатурен номер
- 2) Дата с която да бъде извършено сравнението

```
1. Add new product.
2. Change quantity of product.
3. Show all expired products.
4. Show info for product.
5. Exit
Insert option: 3
Insert date to compare /in dd.mm.yyyy format/: 10.07.2019
id: 2
name: Product2
price: 1.23
quantity: 42
date: 15.05.2019
Continue
```

Figure 9

5.5 Извеждане на информацията за зададена стока

При избор на тази опция се извежда информацията за зададената стока (Figure 10):

- 1) Номенклатурен номер
- 2) Наименование на стоката
- 3) Единична цена
- 4) Количество
- 5) Дата на производство на стоката

, при липсваща стока се извежда че тя липсва (Figure 11).

```
1. Add new product.
2. Change quantity of product.
3. Show all expired products.
4. Show info for product.
5. Exit
Insert option: 4
Insert id: 1
id: 1
name: Product
price: 42.69
quantity: 20
date: 13.05.2019
Continue
```

Figure 10

```
1. Add new product.
2. Change quantity of product.
3. Show all expired products.
4. Show info for product.
5. Exit
Insert option: 4
Insert id: 10
Missing product!
Continue
```

Figure 11

5.6 Изход

С опцията за изход се излиза от програмата (Figure 12). Освен приключване на приложението с избор на тази опция се запазват стоките от склада в базата данни.

```
1. Add new product.
2. Change quantity of product.
3. Show all expired products.
4. Show info for product.
5. Exit
Insert option: 5
Continue
(base) └─nikolay@shifu ~/Workspace/tu/tu-pik-2/project <master*>
└─$
```

Figure 12

Съдържание

1.	Задание № 8	2
1.1	Наличностите в складовете се съхраняват в двоични файлове. За всяка стока в склада се пазят следните данни:	2
1.2	Да се извършват следните обработки:.....	2
1.3	Данните да се поддържат в динамична структура - едносвързан списък в оперативната памет на ПК.....	2
1.4	ИЗИСКВАНИЯ КЪМ ОФОРМЛЕНИЕТО	2
2	Описание на използваните модули(функции) – прототип, входно изходни параметри и предназначение	3
2.1	load_products	3
2.2	save_products	3
2.3	find_by_id	3
2.4	delete_by_id	3
2.5	string_to_time	4
2.6	add_new_product.....	4
2.7	print_product_with_id.....	4
2.8	print_all_expired_products	4
2.9	change_quantity_of_product	5
3	Общо описание за функциониране на програмата (вход/изход).....	6
4	Изходен код на програмата	7
5	Резултати от изпълнението на програмата (контролен пример)	15
5.1	Зареждане на базата данни от файл	15
5.2	Добавяне на нова стока в склада.....	15
5.3	Промяна на наличността от зададена стока	15
5.4	Извеждане на стоките с изтекъл срок на годност	17
5.5	Извеждане на информацията за зададена стока	18
5.6	Изход	19