**Факултет „Автоматика“**

**Курсов проект по ПИК-2**

**Изготвил:**

**Михаил Узунов**

**Факултет Автоматика, група 1 ф No. 011215064**

***Дата: 06.04.2016г. Проверил: ................***

**Съдържание**

1. **Задание..........................................................................................................**
2. **Описание на функциите...............................................................................**

***Да се разработи програма за поддържане на складово стопанство със следните изисквания:***

1.     Данните за наличностите в склада се съхраняват в двоичен файл. За всяка стока в склада се пазят следните данни:

1. Идентификационен номер - уникално 10 цифрено число
2. Наименование на стоката - до 60 символен низ
3. Единична цена - реално число
4. Количество - цяло число
5. Дата на производство на стоката - записана във формата ДД.ММ.ГГГГ
6. Срок на годност на стоката (в дни) - цяло число
7. Име на производителя - до 40 символен низ
8. Данъчен номер на производителя - във формат xxxxxxxxxx ( x – десетична цифра )
9. Изборът на обработка на информация става от прозорец-меню. За целта да се разработят собствени функции за отваряне и затваряне на прозорец, извеждане на текст на зададено място от екрана с зададен цвят и фон, за използуване на клавишите ‘↑’ и ‘↓’ от страничната клавиатура и всички други необходими за целта. Забранява се използуването на готовите библиотечни функции, напр. window(),gotoxy(), settextcolor(), setbkcolor() и др. подобни.
10. Да се извършват следните обработки:   
    а) Да се добавя нова стока в склада.  
    б) Да се променя броя на наличностите от зададена стока ( добавяне или изваждане на количества от склада ). При изваждане, ако заявката е за повече от съществуващото количество - да не се изпълнява, а да се издава съобщение. Ако наличностите станат равни на 0, стоката да се изключи от склада.  
    в) Да може да се извежда информация за всички стоки от склада с изтекъл срок на годност (текущата дата се задава от потребителя).  
    г) Да може да се направи справка за всички производители, подредени в низходящ ред по общата стойност на наличните стоки, доставени от тях. Ако за двама производители има еднаква стойност на стоките, първи да се изпише този, който е доставил по-голям асортимент от стоки. Информацията да се визуализира в текстов прозорец (виж изискванията от т.2) във вида:  
      
     Данъчен номер Име на производителя Брой стоки в асортимента Обща стойност
11. По зададен идентификационен номер да се разпечатва информацията за доставена стока.
12. Да може да се прави справка за всички налични стоки.
13. При стартиране на програмата с файл, подаден през командния ред да могат да се задават цвета на буквите, цвета на фона и цвета на курсора за различните прозорци. Информацията във файла да е подредена в средния ред следния:  
     • номер на цвета на буквата за прозореца-меню  
     • номер на цвета на фона за прозореца-меню  
     • номер на цвета на курсора за прозореца-меню  
     • номер на цвета на буквата за текстовия прозорец  
     • номер на цвета на фона за текстовия прозорец  
    Ако на командния ред не е зададено името на файл, да се използуват цветове по подразбиране.

**Типове данни:**

* **Vector – списък (динамичен масив), който се оразмерява сам, ако се опитва да се добави нов елемент, но няма място в паметта.**
* **ColorConfig – структура да пази моментните стойности за цветове**
* **DateTime – структура за пазене на време във програмата**
* **Merchandise – модел на стоката**
* **Producer – модел за производителите**

**Описание на функциите**

***int main()***

**Входни данни:**

**Няма.**

**Изходни данни:**

**Връща тип integer, резултат на изпълнение на програмата.**

**Декларирани променливи във main():**

* **should\_run (*int)*  - флаг за изход на програмата**
* **selectedMenuOption *(int)*  - моментно избраната опция от менюто**
* **menu\_layer *(int)* – в кой слой на менюто се намира потребителят**
* **text\_rendered *(int)* – флаг за изписване на менюто**
* **colorconf *(ColorConfig)* – съдържа моментната конфигурация за цветовете**
* **merch\_list *(Vector)*  - списък със всички стоки във склада.**
* **ch *(char)*  - моментно въведеният клавиш от клавиатурата**

**Структура на функцията main():**

**Създава се цикъл с флаг should\_run. Принтира се менюто на програмата и се иска от потребителя да избере операцията която иска да извърши програмата.**

***void vector\_init(Vector\* vector)***

**Входни данни:**

**vector – Указател към списък**

**Изходни данни:**

**Няма.**

**Декларирани променливи във vector\_init(Vector\* vector):**

**Няма.**

**Структура на функцията vector\_init(Vector\* vector):**

**Инициализира се нов списък, като на подадения указател му се зададе големина 0, капацитет от INITIAL\_CAPACITY (константа от 16) и се задели памет от тип Merchandise за 16 елемента.**

***void vector\_append(Vector\* vector, struct Merchandise\* value)***

**Входни данни:**

**vector – Указател към списък**

**value – Указател към елемента който трябва да се добави**

**Изходни данни:**

**Няма.**

**Декларирани променливи във vector\_append(Vector\* vector, struct Merchandise\* value):**

**Няма.**

**Структура на функцията vector\_append(Vector\* vector, struct Merchandise\* value):**

**За да се добави нов елемент в списък, първо се проверява дали има място, ако има се добавя елемента, в противен случай се оразмерява двойно по-голям и след това го добавя.**

***struct Merchandise vector\_get(Vector\* vector, int index)***

**Входни данни:**

**vector – Указател към списък**

**index – Индекс на елемента, който трябва да се вземе**

**Изходни данни:**

**Връща намерения елемент от списъка.**

**Декларирани променливи във vector\_get(Vector\* vector, int index):**

**Няма.**

**Структура на функцията vector\_get(Vector\* vector, int index):**

**Проверява се първо дали индекса е в рамките на масива на списъка. Ако е в границите му, се извежда елемента, в противен случай се излиза от програмата с грешка 1.**

***void vector\_resize\_if\_full(Vector\* vector)***

**Входни данни:**

**vector – Указател към списък**

**Изходни данни:**

**Няма.**

**Декларирани променливи във vector\_get(Vector\* vector, int index):**

**Няма.**

**Структура на функцията vector\_get(Vector\* vector, int index):**

**Проверява се дали броя на елементите е равен на капацитета, ако е равен, тогава се уголемява масива двойно.**

***void vector\_free(Vector\* vector)***

**Входни данни:**

**vector – Указател към списък**

**Изходни данни:**

**Няма.**

**Декларирани променливи във vector\_get(Vector\* vector, int index):**

**Няма.**

**Структура на функцията vector\_get(Vector\* vector, int index):**

**Изиква се функцията free(); за да се освободи паметта заделена от списъка.**

***void print\_menu(int menu, int selected, struct ColorConfig\* cfg)***

**Входни данни:**

**menu - Посочва се менюто за принтиране**

**selected – Кой ред е маркиран от потребителя**

**cfg – Конфигурация за цветовете**

**Изходни данни:**

**Няма.**

**Декларирани променливи във print\_menu(int menu, int selected, struct ColorConfig\* cfg):**

**Всички надписи за менютата за изнесени на променливи**

**Структура на функцията print\_menu(int menu, int selected, struct ColorConfig\* cfg):**

**Ако menu е 0 се принтира главното меню, ако е 1 се принтира менюто за мениджниране на стоките в склада, ако е 2 се принтира менюто за избиране на цветове.**

***void clear\_cons()***

**Входни данни:**

**Няма.**

**Изходни данни:**

**Няма.**

**Декларирани променливи във clear\_cons():**

**Няма.**

**Структура на функцията clear\_cons():**

**Извиква се функцията system(“cls”);**

***int get\_max\_options(int menuLayer)***

**Входни данни:**

**menuLayer – слой на менюто**

**Изходни данни:**

**Връща число – броят на елементи в мен**юто

**Декларирани променливи във get\_max\_options(int menuLayer):**

**Няма.**

**Структура на функцията get\_max\_options(int menuLayer):**

**Връща брой на опции в менюто в зависимост от слоя който е подаден.**

***int fix\_for\_up(int selectedOp, int max)***

**Входни данни:**

**selectedOp – Избраният ред от менюто**

**max – последния ред от менюто**

**Изходни данни:**

**Връща число изчислено за правилно избиране на редовете**

**Декларирани променливи във fix\_for\_up(int selectedOp, int max):**

**Няма.**

**Структура на функцията fix\_for\_up(int selectedOp, int max):**

**Ако е избран най-горния ред, при следващото натискане на стрелка нагоре ще избере най-долния ред.**

***int fix\_for\_down(int selectedOp, int max)***

**Входни данни:**

**selectedOp – Избраният ред от менюто**

**max – последния ред от менюто**

**Изходни данни:**

**Връща число изчислено за правилно избиране на редовете**

**Декларирани променливи във fix\_for\_down(int selectedOp, int max):**

**Няма.**

**Структура на функцията fix\_for\_down(int selectedOp, int max):**

**Ако е избран най-долния ред, при следващото натискане на стрелка надолу ще избере най-горния ред.**

***void set\_text\_color(WORD color, char\* text)***

**Входни данни:**

**color – комбинация от цветове**

**text – текстът който трябва да се оцвети**

**Изходни данни:**

**Няма.**

**Декларирани променливи във set\_text\_color(WORD color, char\* text):**

**h – дръжка за стандартния изход на конзолата от Windows**

**wOldColorAttrs – запазват се старите атрибути за цветове**

**consoleInfo – запазва се информация за конзолата**

**Структура на функцията set\_text\_color(WORD color, char\* text):**

**Взима се старата информация, записва се оригиналния цвят и се принтира желаният от потребителя цвят. След това се връща стария цвят на командния прозорец.**

***struct ColorConfig load\_default\_colors()***

**Входни данни:**

**Няма.**

**Изходни данни:**

**Връща конфигурация за цветове по подразбиране**

**Декларирани променливи във load\_default\_colors():**

**Няма.**

**Структура на функцията load\_default\_colors():**

**Зарежда променлива със цветове по подразбиране и я връща.**

***void load\_custom\_colors(struct ColorConfig\* config, char\* path)***

**Входни данни:**

**config – Указател към конфигурацията в която ще се зареждат новите цветове**

**path – име на файла от който ще се чете**

**Изходни данни:**

**Няма.**

**Декларирани променливи във load\_custom\_colors(struct ColorConfig\* config, char\* path):**

**Няма.**

**Структура на функцията load\_custom\_colors(struct ColorConfig\* config, char\* path):**

**Отваря се подаденият файл за четене. Прочита се структурата от двоичния файл и се записва в буфер, от който се записват цветовете в указателя.**

***int save\_to\_disk(Vector\* products)***

**Входни данни:**

**products – Указател към списъка със всички продукти**

**Изходни данни:**

**Връща резултат дали е успяло да запише информацията на диска.**

**Декларирани променливи във save\_to\_disk(Vector\* products):**

**Няма.**

**Структура на функцията save\_to\_disk(Vector\* products):**

**Отваря се файлът database.bin и се записват данните от списъка.**

***int is\_unique(int id, Vector\* products)***

**Входни данни:**

**id – идентификационен номер на продукта**

**products – Указател към списъка със всички продукти**

**Изходни данни:**

**Връща резултат дали идентифационния номер е уникален или не**

**Декларирани променливи във is\_unique(int id, Vector\* products):**

**Няма.**

**Структура на функцията is\_unique(int id, Vector\* products):**

**Минава се през всички елементи от списъка с продукти и се проверя дали подаденият идентификационен номер присъства във някой от продуктите.**

***struct Merchandise make\_new\_merchandise(struct Vector\* list)***

**Входни данни:**

**list – Указател към списъка със всички продукти**

**Изходни данни:**

**Създава нов обект Merchandise и го връща.**

**Декларирани променливи във make\_new\_merchandise(struct Vector\* list):**

**Няма.**

**Структура на функцията make\_new\_merchandise(struct Vector\* list):**

**За всички членов на Merchandise се взима вход данни от потребителя и се записва във нов обект Merchandise.**

***void get\_time\_by\_string(struct DateTime\* date, char\* date\_string)***

**Входни данни:**

**date – Указател към дата.**

**date\_string – Форматирана дата за четене**

**Изходни данни:**

**Няма.**

**Декларирани променливи във get\_time\_by\_string(struct DateTime\* date, char\* date\_string):**

**Няма.**

**Структура на функцията get\_time\_by\_string(struct DateTime\* date, char\* date\_string):**

**Разбива подадения стринг на различни части, като ги разделя със „:“ и записва числата на съответния член в указателя към датата.**

***void get\_empty\_input()***

**Входни данни:**

**Няма.**

**Изходни данни:**

**Няма.**

**Декларирани променливи във get\_empty\_input():**

**Няма.**

**Структура на функцията get\_empty\_input():**

**Пуска се цикъл който чете един символ за да изчисти потока от вход от конзолата.**

***void print\_all\_values(Vector\* merch\_list)***

**Входни данни:**

**Няма.**

**Изходни данни:**

**Няма.**

**Декларирани променливи във print\_all\_values(Vector\* merch\_list):**

**Няма.**

**Структура на функцията print\_all\_values(Vector\* merch\_list):**

**Итерира се през всички елементи от списъка със продукти и се принтират техните стойности.**

***void print\_char(int count, char ch)***

**Входни данни:**

**count – Колко на брой символа трябва да се принтират.**

**ch – Кой символ трябва да се принтира**

**Изходни данни:**

**Няма.**

**Декларирани променливи във print\_char(int count, char ch):**

**Няма.**

**Структура на функцията print\_char(int count, char ch):**

**Принтират се count на брой символи ch, чрез for цикъл.**

***void print\_product\_by\_id(Vector\* list, int id)***

**Входни данни:**

**list – Указател към списъка със всички елементи**

**id – Идентификационен номер на продукт от списъка**

**Изходни данни:**

**Няма.**

**Декларирани променливи във print\_product\_by\_id(Vector\* list, int id):**

**Няма.**

**Структура на функцията print\_product\_by\_id(Vector\* list, int id):**

**Намира се елемента на който подадения идентификационен номер е равен на някой от елементите в списъка, и се извика функцията print\_single\_product();**

***void print\_single\_product (struct Merchandise product)***

**Входни данни:**

**product – Указател към продукта, който трябва да се принтира.**

**Изходни данни:**

**Няма.**

**Декларирани променливи във print\_single\_product(struct Merchandise product):**

**Няма.**

**Структура на функцията print\_single\_product(struct Merchandise product):**

**Принтират се всички членове от структурата.**

***void print\_all\_expired\_products (Vector\* list)***

**Входни данни:**

**list – Указател към списъка със всички продукти**

**Изходни данни:**

**Няма.**

**Декларирани променливи във print\_all\_expired\_products(Vector\* list):**

**Няма.**

**Структура на функцията print\_all\_expired\_products(Vector\* list):**

**Итерират се всички елементи от списъка и се проверява дали срока е изтекъл, чрез функцията has\_expired() и се принтират всички членове от структурата Merchandise**

***int has\_expired(struct Merchandise product)***

**Входни данни:**

**product – Продукта който трябва да се проверява**

**Изходни данни:**

**Връща резултат дали е с изтекъл срок или не**

**Декларирани променливи във has\_expired(struct Merchandise product):**

**Няма.**

**Структура на функцията has\_expired(struct Merchandise product):**

**Проверява се колко дни са минали от датата на създаване на продукта и сегашната дата, ако дните са повече от срока на годност, продукта е с изтекъл срок.**

***void cpy\_time(struct DateTime\* time, struct tm\* str\_time)***

**Входни данни:**

**time – Датата от която ще се копира**

**str\_time – датата в която ще се копира**

**Изходни данни:**

**Няма.**

**Декларирани променливи във cpy\_time(struct DateTime\* time, struct tm\* str\_time):**

**Няма.**

**Структура на функцията cpy\_time(struct DateTime\* time, struct tm\* str\_time):**

**Копира всеки член от time в str\_time.**

***int get\_days\_in\_months(int months)***

**Входни данни:**

**months – Датата от която ще се копира**

**Изходни данни:**

**Няма.**

**Декларирани променливи във get\_days\_in\_months(int months):**

**Няма.**

**Структура на функцията get\_days\_in\_months(int months):**

**Намира колко дни има в months на брой месеца.**

***void edit\_by\_id(int id, Vector\* list, Vector\* temp)***

**Входни данни:**

**id – Идентификационен номер**

**list – Списък със всички елементи**

**temp – Временен списък за изтриване на елементите от диска**

**Изходни данни:**

**Няма.**

**Декларирани променливи във edit\_by\_id(int id, Vector\* list, Vector\* temp):**

**Няма.**

**Структура на функцията edit\_by\_id(int id, Vector\* list, Vector\* temp)**

**Намира продукта с подадения идентификационен номер от потребителя. След това пита потребителя какво иска да направи, ако иска да добави стока, се въвежда броя и прибавя към стоката. Ако иска да извади, не може да вади повече отколкото има в склада, и ако броя стане 0, се премахва стоката от диска.**

***void sort\_producers(Vector\* list)***

**Входни данни:**

**list – Списък със всички продукти**

**Изходни данни:**

**Няма.**

**Декларирани променливи във sort\_producers(Vector\* list):**

**Няма.**

**Структура на функцията sort\_producers(Vector\* list):**

**Първо се намират всички прозиводители без повторения с техния брой стоки в асортимента и общ брой доставени стоки. След това се сортират в низходящ ред. Като се минава елемент по елементи и ако сегашния елемент е по-малък от следващия се разменят позициите със функцията swap\_producers().**

***int check\_if\_exists (Producer\* producers, int size, char\* curr\_name)***

**Входни данни:**

**producers – Списък със всички производители**

**size – колко на брой производители има в списъка**

**curr\_name – името на производителя който се търси**

**Изходни данни:**

**Връща резултат дали е намерен производителя в масив.**

**Декларирани променливи във check\_if\_exists(Producer\* producers, int size, char\* curr\_name):**

**Няма.**

**Структура на функцията check\_if\_exists(Producer\* producers, int size, char\* curr\_name):**

**Минава се през всеки елемент от масива с производителите и се сравнява името. Ако е едно и също с някой от елементите значи съществува, в противен случай не.**

***void swap\_producers(Producer\* a, Producer\* b)***

**Входни данни:**

**a – Производител а**

**b – Производител б**

**Изходни данни:**

**Няма.**

**Декларирани променливи във get\_days\_in\_months(int months):**

**Няма.**

**Структура на функцията get\_days\_in\_months(int months):**

**Разменят се стойностите на двете променливи.**