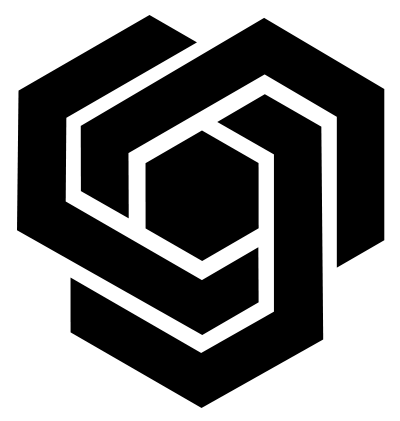
**ТЕХНИЧЕСКИ УНИВЕРСИТЕТ СОФИЯ**



Курсова работа

ПИК II

**Изработил:**

**Проверил:**

**Задача:**

**Да се състави програма тип „меню” за поддръжка на каталог за автокъща със следните изисквания:**

1. За колите в една автокъща се съхраняват следните данни:
2. идентификационен код (букви и цифри);
3. цвят – текст (20 символа);
4. марка – текст 20 символа;
5. цена - реална стойност.
6. Информацията се съхранява в подходяща файлова структура, като се осигурят следните функции:

а) Въвеждане на данни за нови коли в подходящ диалогов прозорец;

б) Търсене и извеждане на колите по даден цвят;

в) Изтриване на кола с посочен код, като се запази подредеността на списъка;

г) Намаляване с 10 процента цените на колите от дадена марка.

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

**Решение:**

Програмата се реализира чрез 12 функции, главна main и още 9 - insertLAST; last; ADDinLAST; readFILE; removeCAR; menu; PRINTcarSEARCH;PRICEcar;writeFILE; MEMORYfree, char markINPUTр char colorINPUT,които са съответно за:създаване на указател към свързан списък чрез четене от файл;

създаване на указател към свързан списък чрез конзола,

четене на данните от файл, Изтриване на кола с посочен код, като се запази подредеността на списъка,меню,търсене на дадена кола по зададен цвят ,функция за отстъпка с 10 процента по дадена марка,записване в зададен двойчен файл,освобождаване на списъка.

В главната функция се извиква функцията menu(), връщаща избора на потребителя, който се записва в локалната променлива result. В зависимост от стойността на result в main се извиква съответната функция, извършваща избраното от потребителя действие.

**функция int main()**

Не приема променливи, но връща променлива от тип int, която представлява избора на потребителя.

int key - променлива от тип int, на която се присвоява избора на потребителя, направен във функцията menu

carMEMORY \*head = NULL - променлива от тип carMEMORY, която да бъде начало (глава) на списъка

carIN carTemp - променлива от тип carIN, чрез която да запишем данните от файла в списък

**функция carMEMORY \*insertLAST(carMEMORY \*\*head,carIN info);**

Приема двоен указател от тип carMEMORY и променлива от тип carIN и връща адреса на \*head.

carMEMORY \*temp1 - указател от тип carMEMORY, чрез който се създава списъка и carMEMORY \*lastp1 ,който се приравнява на указателя сочещ съм опашката на списъка.

**Функция carMEMORY \*last(carMEMORY \*\*head);**

Приема двоен указател от тип carMEMORY и връща указател temp към адреса на структората ,която съдържа адреса на опашката NULL.

**функция carMEMORY \*ADDinLAST(carMEMORY \*\*head);**

Приема двоен указател от тип carMEMORY и променливи от тип carMEMORY,като и 2 променливи от тип int и влъща указател към главата на списъка.

**функция readFILE(carMEMORY \*\*head,carIN infoTEMP);**

Приема двоен указател от тип carMEMORY и променлива от тип carIN и не връща нищо.

FILE \*fp - файлов указател, чрез който да се достигне до отваряния файл.

За, да запише информацията от файла в списък, извиква функцията insertLAST, чрез която се създава списъка.

**функция void removeCAR(carMEMORY \*\*head);**

Приема указател от тип carMEMORY и не връща нищо.

carMEMORY \*temp2 - указател от тип carMEMORY, чрез който се обхожда списъка

**функция**

**carMEMORY \*PRINTcarSEARCH(carMEMORY \*\*head,char\* colour);**

Приема указател от тип carMEMORY и не връща нищо.

carMEMORY \*temp3 - указател от тип carMEMORY, чрез който се обхожда списъка.

int true - проверка дали има съвпадения.

char buffer1- използва се за запис на въведената от потребителя информация.

**функция double PRICEcar(carMEMORY \*\*head, char\* Tmark);**

Приема указател от тип carMEMORY и връща double .

carMEMORY \*temp4 - указател от тип carMEMORY, чрез който се обхожда списъка.

int true - проверка дали има съвпадения.

char buffer2- използва се за запис на въведената от потребителя информация.

tempOffPrice – намаление 10 процента

**функция void writeFILE(carMEMORY \*\*head);**

Приема указател от тип carMEMORY и не връща нищо.

carMEMORY \*temp4 - указател от тип carMEMORY, чрез който се обхожда списъка

FILE \*fp - файлов указател, за достигане до файла

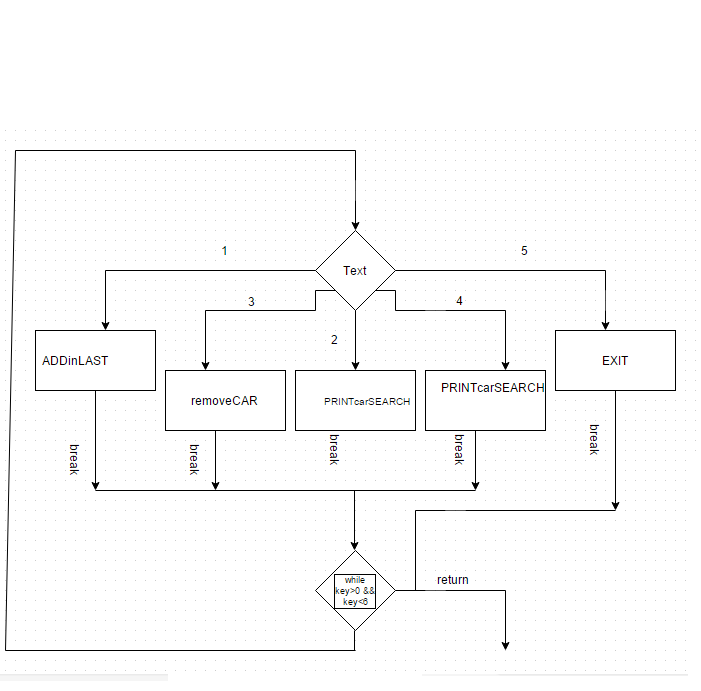
**Функция void MEMORYfree(carMEMORY \*\*head);**

Приема двоен указател от тип carMEMORY и не връща нищо.

carMEMORY \*temp7 - указател от тип carMEMORY, чрез който се обхожда списъка и ,чрез който временно се запазва указателят на елемента, който се изтрива.

**функция int menu()**

Не приема променливи, но връща променлива от тип int, която е избора на потребителя.

****

int result - променлива от тип int, чрез която се записва избора на потребителя.

**Блок схема main:**

**Source code:**

#ifndef HEADERFINAL\_H\_INCLUDED

#define HEADERFINAL\_H\_INCLUDED

typedef struct input

{

char code1[20];

char color1[20];

char mark1[20];

double price1;

}carIN;

typedef struct car

{

char code[20];

char color[20];

char mark[20];

double price;

struct car \*pointer;

}carMEMORY;

int menu();

carMEMORY \*FILEinsertLAST(carMEMORY \*\*head, carIN info);

carMEMORY \*last(carMEMORY \*\*head);

carMEMORY \*ADDinLAST(carMEMORY \*\*head);

void readFILE(carMEMORY \*\*head);

void removeCAR(carMEMORY \*\*head);

carMEMORY \*PRINTcarSEARCH(carMEMORY \*\*head, char TEMPcolor2[20]);

void PRICEcar(carMEMORY \*\*head, char TEMPmark1[20]);

void writeFILE(carMEMORY \*\*head);

void MEMORYfree(carMEMORY \*\*head);

void colorINPUT(char Tcolor[20]);

void markINPUT(char Tmark[20]);

#endif // HEADERFINAL\_H\_INCLUDED

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "headerFinal.h"

int main()

{

int key;

carMEMORY \*head = NULL;

readFILE(&head);

do

{

key = menu();

switch (key)

{

case 1:

{

ADDinLAST(&head);

break;

}

case 2:

{

char TEMPcolor[20];

colorINPUT(TEMPcolor);

PRINTcarSEARCH(&head, TEMPcolor);

break;

}

case 3:

{

removeCAR(&head);

break;

}

case 4:

{

char TEMPmark[20];

markINPUT(TEMPmark);

PRICEcar(&head, TEMPmark);

break;

}

case 5:

{

writeFILE(&head);

MEMORYfree(&head);

printf("\n\n \_-\_-Goodbye!-\_-\_\n\n");

break;

}

}

} while (key != 5);

return 0;

}

int menu()

{

int result;

printf(" MENU \n");

printf("ENTER 1 to add new cars\n");

printf("ENTER 2 to view the cars in the list by color \n");

printf("ENTER 3 to delete car from the list by ID \n");

printf("ENTER 4 to 10 percent off by mark\n");

printf("ENTER 5 to EXIT\n");

scanf("%d", &result);

return result;

}

carMEMORY \*ADDinLAST(carMEMORY \*\*head)

{

int n, i = 0;

printf("Enter how many cars you want to enter\n");

scanf("%d", &n);

for (i = 0;i<n;i++)

{

carMEMORY \*temp = (carMEMORY\*)malloc(sizeof(carMEMORY));

carMEMORY \*lastp;

if (temp == NULL)

printf("Allocation ERROR \n");

printf("\n");

printf("Enter ID: ");

scanf("%s", temp->code);

fflush(stdin);

printf("Enter color: ");

fflush(stdin);

scanf("%s", temp->color);

fflush(stdin);

printf("Enter mark: ");

scanf("%s", temp->mark);

fflush(stdin);

printf("Enter price: ");

scanf("%lf", &(temp->price));

fflush(stdin);

printf("\n");

if (\*head == NULL)

{

\*head = temp;

temp->pointer = NULL;

}

else

{

lastp = last(head);

lastp->pointer = temp;

temp->pointer = NULL;

}

}

return \*head;

}

carMEMORY \*last(carMEMORY \*\*head)

{

carMEMORY \*temp;

for (temp = \*head;temp->pointer != NULL;temp = temp->pointer)

;

return temp;

}

void readFILE(carMEMORY \*\*head)

{

carIN infoTEMP;

FILE \*fp;

fp = fopen("carAuto.txt", "rb");

if (fp != NULL)

{

for (;!feof(fp);)

{

fread(infoTEMP.code1, sizeof(char) \* 20, 1, fp);

fread(infoTEMP.color1, sizeof(char) \* 20, 1, fp);

fread(infoTEMP.mark1, sizeof(char) \* 20, 1, fp);

fread(&(infoTEMP.price1), sizeof(double), 1, fp);

FILEinsertLAST(head, infoTEMP);

}

}

if (\*head != NULL)

{

\*head = (\*head)->pointer;

}

//fclose(fp);

}

carMEMORY \*FILEinsertLAST(carMEMORY \*\*head, carIN info)

{

carMEMORY \*temp1 = (carMEMORY\*)malloc(sizeof(carMEMORY));

carMEMORY \*lastp1;

if (temp1 == NULL)

printf("Allocation ERROR \n");

strcpy(temp1->code, info.code1);

strcpy(temp1->color, info.color1);

strcpy(temp1->mark, info.mark1);

temp1->price = info.price1;

if (\*head == NULL)

{

\*head = temp1;

temp1->pointer = NULL;

}

else

{

lastp1 = last(head);

lastp1->pointer = temp1;

temp1->pointer = NULL;

}

return \*head;

}

void removeCAR(carMEMORY \*\*head)

{

int g = 0, nn;

char buffer[20];

carMEMORY \*temp3 = \*head;

carMEMORY \*p;

if (\*head != NULL)

{

printf("How many cars you want to remove: \n");

scanf("%d", &nn);

for (g = 0;g<nn; g++)

{

printf("Enter car ID: \n");

scanf("%s", buffer);

do

{

if (!strcmp(buffer, temp3->code))

{

if (temp3 == \*head)

{

if ((\*head)->pointer != NULL)

{

\*head = (\*head)->pointer;

free(temp3);

printf("list deleted \n");

break;

}

else

{

\*head = NULL;

free(temp3);

printf("list deleted \n");

break;

}

}

else if (temp3->pointer == NULL)

{

p->pointer = NULL;

free(temp3);

temp3 = \*head;

printf("list deleted \n");

break;

}

else

{

p->pointer = temp3->pointer;

free(temp3);

temp3 = \*head;

printf("list deleted \n");

break;

}

}

if (temp3->pointer == NULL)

printf("NO CAR WITH THAT ID \n");

p = temp3;

temp3 = temp3->pointer;

} while (temp3 != NULL);

buffer[0] = '\0';

}

}

else

printf("LIST EMPTY \n");

}

void colorINPUT(char Tcolor[20])

{

printf("Enter car color \n");

scanf("%s", Tcolor);

}

void markINPUT(char Tmark[20])

{

printf("Enter car mark \n");

scanf("%s", Tmark);

}

carMEMORY \*PRINTcarSEARCH(carMEMORY \*\*head, char TEMPcolor2[20])

{

carMEMORY \*temp3;

int true = 0;

if (\*head != NULL)

{

for (temp3 = \*head;temp3 != NULL;temp3 = temp3->pointer)

{

if (!strcmp(TEMPcolor2, temp3->color ))

{

printf(" %s %s %s %.2lf \n\n", temp3->code, temp3->color, temp3->mark, temp3->price);

true = 1;

}

}

if (true == 0)

{

printf("There is not such a car with that mark! \n ");

}

}

else

printf("list in empty!");

return NULL;

}

void PRICEcar(carMEMORY \*\*head, char TEMPmark1[20])

{

carMEMORY \*temp4;

int true = 0;

double tempOffPrice = 0;

if (\*head != NULL)

{

for (temp4 = \*head;temp4 != NULL;temp4 = temp4->pointer)

{

if (!strcmp(TEMPmark1, temp4->mark))

{

tempOffPrice = temp4->price / 10;

temp4->price = temp4->price - tempOffPrice;

true = 1;

}

}

if (true == 0)

{

printf("There is not such a car with that mark! \n ");

}

}

else

printf("list in empty!");

printf("\n");

}

void writeFILE(carMEMORY \*\*head)

{

FILE \*fp1;

fp1 = fopen("carAuto.txt", "wb");

carMEMORY \*temp4;

if (\*head != NULL)

{

for (temp4 = \*head;temp4 != NULL;temp4 = temp4->pointer)

{

fwrite(temp4->code, sizeof(char) \* 20, 1, fp1);

fwrite(temp4->color, sizeof(char) \* 20, 1, fp1);

fwrite(temp4->mark, sizeof(char) \* 20, 1, fp1);

fwrite(&(temp4->price), sizeof(double), 1, fp1);

}

fclose(fp1);

printf("list saved!! \n\n");

}

else

printf("Error with the list \n");

}

void MEMORYfree(carMEMORY \*\*head)

{

carMEMORY \*temp7 = NULL;

for (;temp7 != NULL; temp7 = \*head)

{

\*head = temp7->pointer;

free(temp7);

}

\*head = NULL;

}