Министерство образования Республики Беларусь
Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет непрерывного и дистанционного обучения

Кафедра экономической информатики

Контрольная работа №1
По курсу «Основы алгоритмизации и программирования» часть 2

Выполнил — Студент
Ф.И.О. Юруш Е.В.
№ зач. кн. (нет)

Проверил — Т. М. Унучек

Минск-2017

Содержание

Введение

Выполнение данной контрольной работы заключается в разработке консольного приложения для автоматизации учёта продаж программного обеспечения с использованием языка программирование Си, Её цель - овладеть начальными практическими навыками программирования на языке Си и работы в среде программирования Visual C++, разбор структуры программы и построение её алгоритма, формирование компьютерной грамотности.

1. Теоретические вопросы

Файлы. Двоичное и текстовое представление файлов.

Файлом является набор данных на внешнем носителе, который рассматривается в процессе обработки как единое целое. Файл может быть текстовый и двоичный (бинарный).

Текстовый файл – последовательность ASCII - символов. Данный тип файла позволяет хранить информацию в виде понятном для человека. Однако, каждый символ занимает в памяти один байт, что увеличивает размер файла, по сравнению с хранением информации в двоичном виде, а также при каждом считывании числа из текстового файла или записи происходит трансформация числа в строку или обратно, что затратно. Вышеописанные нюансы можно избежать используя двоичный файл для хранения информации.

Двоичный файл – это последовательность данных, структура которых определяется программно; неограниченный массив байт. В данном типе файла любая переменная занимает фиксированное количество байтов, которое определяется ее типом. Данные в файл переносятся без преобразований, а доступ к данному файлу может быть произвольным, т.е. обращаться можно непосредственно к любой его части.

Использование очередей при реализации запросов ввода-вывода.

Использование очередей при реализации запросов ввода информации конструируется следующим образом. При вводе информации пользователем, она, прежде чем записаться в место конечного хранения, попадает в очередь. В самой очереди, с ней можно выполнять такие операции как редактирование, удаление из очереди, просмотр, при этом она всё ещё не будет записана по конечному адресу. Извлечение её из очереди будет происходить в порядке её попадания в неё, таким образом при записи информации из очереди по конечному адресу, порядок ввода не будет нарушен. Извлечение информации из очереди и её запись по необходимому адресу может быть реализовано при переполнении очереди, при очистки очереди (отправкой всех данных на

запись), либо при операции принудительного извлечения информации из очереди в порядке её следования. Очередь при операции запросов вывода реализуется по такому же принципу.

2. Практическая часть

Задание на разработку программы в соответствии с индивидуальным заданием: в соответствии с вариантом индивидуального задания, необходимо разработать консольное приложение на языке С. Приложение должно предоставлять возможности: просмотра информации из текстового файла; добавления новых записей в файл; удаления записей из файла; редактирования записей в файле. В работе предусмотреть использование пользовательских функций, массивов, структур, динамического распределения памяти, поиск информации по условию, вводимому пользователем с клавиатуры, сортировку (не менее 3-х способов).

Вариант индивидуального задания 1, предметная область «Учет продаж программного обеспечения.».

Листинг кода программы:

Создаём заголовочный файл header.h. В нём:

```
//объявляем константу в которой храним количество символов
//в поле название структуры Merchant
#define MAXFNAME 25
```

```
//объявляем структуру
typedef struct tagMerchant
{
    int    merchNmb;            //номер в списке
    char   fname[MAXFNAME];     //название товара
    int    price;               //цена товара
    int    sold;                //количество проданных позиций

    //в этих переменных храним ссылку на следующую, предыдущую связанную структуру
    struct tagMerchant *pnext, *pprev;
} Merchant;
```

```
17  //объявляем прототипы функций используемых в программе
18  void print(Merchant *p);
19  void getUser(Merchant *p);
20  Merchant* addEnd(Merchant *p, Merchant *end, int stnumb);
21  void printList(Merchant *p);
22  void loadList(Merchant *p, char *file, Merchant **pend, Merchant **pbegin);
23  void saveList(Merchant *p, char *file);
24  int editStd(Merchant *p, Merchant **pend, Merchant **pbegin);
25  int deleteStd(Merchant *p, Merchant **pend, Merchant **pbegin);
26  void find(Merchant *p);
27  Merchant* insInv(Merchant *p1, Merchant *p2);
28  void insSort(Merchant *p, Merchant **begin, Merchant **end);
29  void signSort(Merchant *p, Merchant **begin, Merchant **end);
30  void exchSort(Merchant *p, Merchant **begin, Merchant **end);
```

Создаём главный файл main.c. В нём:

```
1  //подключаем необходимые заголовочные файлы стандартной библиотеки
2  #include<stdio.h>
3  #include<stdlib.h>
4  #include<string.h>
5  //подключаем созданный заголовочный файл
6  #include "header.h"
```

```
8  //пишем главную функцию программы
9  void main()
10 {
11     //объявляем переменные
12     //переменная для выбранного пользователем номера команды
13     int n=0;
14     //перем. для временного хранения стр-ры, для передачи её из одной функции в другую
15     Merchant std;
16     //указатели на первый и последний элемент связанных структур
17     Merchant *begin=NULL, *end=NULL;
```

Организуем пользовательское меню.

```
    //пользовательское меню
    //1.добавить запись в список товаров 2.отредактировать запись в списке
    //3.сохр. список в файл 4.загруз. список из файла 5.вывести список на экран
    //6.выход
L:  printf("\n1.add    3.save   5.print  7.find     9.sortSign\n2.edit\
     4.load   6.exit   8.sortInsr 10.exchSort\n");
    printf("Input the command number (from 1 to 7): ");
    //сохраняем введённое пользователем значение в переменной n
    scanf("%d", &n);

    //в зав-ти от введённого пользователем значения (1-7) запуск. соотв. функции,
    //при вводе '6' (6й команды) выходим без ошибки
    switch(n)
    {
    case 1: //принимаем значения от пользователя во временную структуру std
            getUser(&std);
            //связываем заполненную польз-ем стр-ру std с имеющимися структурами
            end=addEnd(&std,end,0);
            //при первом заполнении стр-ры указатели на 1 и последн. структуру равны
            if (begin==NULL) begin=end;
            break;
    case 2: //редактируем выбранную пользователем строку
            editStd(begin, &end, &begin);
            break;
    case 3: //сохраняем список в файл
            saveList(begin,"list.dat"); break;
    case 4: //загружаем список из файла
            loadList(begin,"list.dat",&end, &begin); break;
    case 5: //печатаем список на экран
            printList (begin); break;
    case 6: //завершаем программу без ошибки
            exit(0); break;
    case 7: //находим необходимую строку
            find(begin); break;
    case 8: //сортируем двусвязный список вставками
            insSort(begin, &begin, &end);
    case 9: //сортируем двусвязный выделением
            signSort(begin, &begin, &end);
    case 10: //сортируем двусвязный список обменом (пузырьковая с-ка)
            exchSort(begin, &begin, &end);
    }
    //при вводе пользователем значения не равного 1,2,3,4,5,6,7
    //просим пользователя ввести команду ещё раз
    goto L;
}
```

Опишем функции которые запускаются при вводе команды добавления записи в список ‘1’.

```c
//функция для приёма значений от пользователя
//вход параметр: указатель на структуру
void getUser(Merchant *p)
{
    Merchant tmp;
    //поочерёдно запрашиваем и принимаем информацию от пользователя
    printf("\nInput merchant :");
    fflush(stdin);
    fgets(tmp.fname,MAXFNAME,stdin);
    //удаляем символ завершения строки кот. добавился функцией fgets
    tmp.fname[strlen(tmp.fname)-1]='\0';
    printf("Input sold amount: ");
    scanf("%d", &tmp.sold);
    fflush(stdin);
    printf("Input price: ");
    scanf("%d", &tmp.price);

    //указатели на пред. след. структуру укажем позже
    tmp.pnext = tmp.pprev = NULL;
    //копируем структуру tmp в структуру по указателю p (вход. пар-р)
    *p = tmp;
}
```

```c
//функция (Ф) для связывания заполненной польз-ем структуры с имеющимся списком
//структур. Ф принимает: заполненную структуру, указатель на последнюю
//структуру в списке, номер в списке который необходимо заполнить для
//добавляемого товара (т.к. данная ф-я еще используется при загрузке
//списка из файла). Ф возвращает: указатель на структуру который записывается в
//переменную end - указатель на последнюю структуру
Merchant* addEnd(Merchant *p, Merchant *end, int stnumb)
{
    //выделяем память для структуры, помещаем указатель на неё в pAdd
    Merchant *pAdd=(Merchant*)malloc(sizeof(Merchant));
    //заполняем память по адресу pAdd переданной, через параметр указатель p, структурой
    *pAdd=*p;
    if (end==NULL) //если добавляем первый элемент
    {
        //в указатель end помещаем указатель на заполненную структуру
        end=pAdd;
        //если переданный параметр stnumb равен нулю, тогда номеру в списке присваиваем 1
        //иначе значение равное переданному параметру stnumb
        if (stnumb==0) pAdd->merchNmb=1;
        else pAdd->merchNmb=stnumb;
    }
    else //если добавляем не первый элемент
    {
        end->pnext = pAdd;//связываем последний элемент с добавляемым
        pAdd->pprev=end;  //связываем добавляемый элемент с последним
        //если переданный параметр stnumb равен нулю, тогда номер в списке присваиваем
        //номер последней строки в списке плюс один
        if (stnumb==0) pAdd->merchNmb=end->merchNmb+1;
        else  pAdd->merchNmb=stnumb;
        end= pAdd;
    }
    //возвращаем указатель на последний элемент
    return end;
}
```

Опишем функции, которые запускаются при вводе команды редактирования записи в списке ‘2’.

```
//функция для редактирования выбранных пользователем строк
//вход. пар-ры: указатель на первую структуру,
//указатель на указатель последней в списке структуры,
//указатель на указатель первой в списке структуры,
int editStd(Merchant *p, Merchant **pend, Merchant **pbegin)
{   //объявляем переменные
    int n, k;
    Merchant *s, std;
    //выводим список на экран и предлагаем выбрать номер строки
    printList (p);
    printf("Input number of Merchant: ");
    scanf("%d", &n);
    //находим структуру номер в списке кот. равен выбранному
    while(p!=NULL) {
    if (p->merchNmb==n){     s=p; break;}
    p=p->pnext; }
    //предлагаем выбрать номер команды "что сделать со строкой"
L:  printf("\nWhat do you want to do with it? \n1.edit \n2.delete\
 \n3.exit to the main menu \nInput the command number : ");
    scanf("%d", &k);
    switch(k)
    {
    case 1: //при редактировании строки запрашиваем перевввести данные
            getUser(&std);//и сохраняем их в структуру std
            //в выбранную пользователем строку вносим изменения
            strncpy(s->fname,std.fname,sizeof(std.fname));
            s->price=std.price; s->sold=std.sold;
            goto M; //выход из цикла
    case 2: //удаляем строку функцией deleteStd, передаём в неё
            //указатель на удаляемую стр-ру, на первый и последний эл-ты списка
            deleteStd(s, pend, pbegin); goto M; //и выходим из цикла
    case 3: goto M; //выход из цикла в главное меню
    }
    goto L;  //переспрашиваем номер команды если введённой значение не 1,2,3
M:  return 0;//выходим без кода об ошибке
}
```

Опишем функцию по указателю deleteStd.

```
158  //удаление строки из списка, параметры идентичные функции editStd
159 int deleteStd(Merchant *p, Merchant **pend, Merchant **pbegin)
160  {
161      Merchant *pr=NULL, *pn=NULL;//указатели для пред. след. эл. списка
162      if (p->pnext==NULL) //если удаляемая строка последняя
163      {
164          pr=p->pprev; //сохраняем в pr указатель на предыд. эл. списка
165          if(pr!=NULL) //если предыд. эл. есть
166          {
167              pr->pnext=NULL; //обнуляем указатель в пред. эл-те
168              *pend=pr;       //предыдущий элемент объявляем последним
169
170          } else //если предыд. элемента нет (т.е. элемент в списке один)
171          {
172              *pend=NULL; //последний и первый элементы списка обнуляем
173              *pbegin=NULL;
174          }
175      }
176      else if (p->pprev==NULL) //если удаляемая строка первая
177      {
178          pn=p->pnext; //сохраняем в pn указатель на след. эл. списка
179          *pbegin=pn;  //указатель на первый элемент меняем на pn
180          pn->pprev=NULL; //в след эл-те обнуляем указатель на пред. эл-т
181      }
182      else //если удаляемая стр. ни первая ни последняя
183      {
184          pn=p->pnext; //сохраняем в pn указатель на след. эл. списка
185          pr=p->pprev; //сохраняем в pr указатель на предыд. эл. списка
186     //связываем указ-ли предыдущ. и след. эл-та так как тек. эл-т удаляем
187          pn->pprev=p->pprev;
188          pr->pnext=p->pnext;
189      }
190      free(p); //освобождаем память по указателю тек. элемента
191      p=NULL;  //обнуляем указатель
192      return 0; //выходим без кода ошибки
193  }
```

Опишем функцию, которая запускается при вводе команды сохранения списка в файл ‘3’.

```
194 //функция записи списка в файл
195 //параметр: указатель на первый элемент списка и название файла
196 void saveList(Merchant *p, char *file)
197 {
198     //объявляем указатель pf типа FILE, открываем/создаём файл
199     //функцией fopen (режим символьн. записи), и помещаем указатель на файл в pf
200     FILE *pf=fopen(file,"w");
201     //если файл успешно открыт/создан
202     if (pf!=NULL)
203     {
204         while(p!=NULL)//проходимся по всем элементам списка
205         {
206         //и записываем элементы структур в файл
207         fprintf(pf,"%d %s %d %d\n", p->merchNmb, p->fname, p->price, p->sold);
208         p=p->pnext;
209         }
210         fclose(pf);//закрываем файл
211     }
212 }
```

Опишем функцию, которая запускается при вводе команды загрузки списка из файла ‘4’.

```
//функция загрузки списка из файла, параметры:
//указатель на структуру типа Merchant, название файла, указатель на
//указ-ль на первый эл-т списка, указ-ль на указ-ль на последний эл-т списка
void loadList(Merchant *p, char *file, Merchant **pend, Merchant **pbegin)
{
    Merchant tmp, *ptmp;//
    //объявляем указатель pf типа FILE, открываем/создаём файл
    //функцией fopen (режим символьн. чтения), и помещаем указатель на файл в pf
    FILE *pf=fopen(file,"r");
    //если файл успешно открыт/создан
    if (pf!=NULL)
    {
        //очищаем текущий список структур
        while(p!=NULL)
        {
            ptmp=p->pnext;
            deleteStd(p, *pend, *pbegin);
            p=ptmp;
        }
        *pend=NULL;
        *pbegin=NULL;
        //пока не достигнут конец файла читаем данные во временную структуру tmp
        while(!feof(pf))
        {
            fscanf(pf,"%d %s %d %d\n", &tmp.merchNmb, &tmp.fname, &tmp.price, &tmp.sold);
            tmp.pnext = tmp.pprev = NULL;
            //запускаем функцию addEnd, передаём в неё указатель на заполненную временную структуру,
            //указатель на указатель последнего элемента списка, и номер записываемого элемента в списке
            //функция добавит полученную из файла структуру в список
            *pend=addEnd(&tmp,*pend,tmp.merchNmb);
            //если элемент первый то последний элемент равен первому элементу
            if (*pbegin==NULL) *pbegin=*pend;
        }
        fclose(pf);//закрываем файл
    }
}
```

Опишем функцию, которая запускается при вводе команды печати списка на экран ‘5’.

```
//функция вывода списка на экран принимает указатель на структуру
void printList(Merchant *p)
{
    //выводим шапку таблицы
    printf("\n%s %s %s %s \n","# ","Merchant                    ","Amount","Price");
    printf("%s \n","-----------------------------------------");

    //циклом проходимся по элементам списка и выводим их на экран
    //ниже описанной функцией print
    while(p!=NULL)
    {
        print(p);
        p=p->pnext;
    }
}
//функция печати строки списка н экран
void print(Merchant *p)
{
    printf("%-2d %-25s %-7d %-5d \n", p->merchNmb, p->fname, p->sold, p->price);
}
```

Опишем функцию, которая запускается при вводе команды поиска товаров по его наименованию.

```
//функция поиска необходимых строк на экран, принимает указатель на первый элемент связанного списка
void find(Merchant *p)
{   //вводим переменную флаг(0,1) и переменную для хранения ввода пользователя
    int  flag=0;
    char tempname[MAXFNAME];
    //запрашиваем имя товара для поиска
    printf("\nInput name of merchant which you want to find:");
    scanf("%s", tempname);
    //циклом проходимся по элементам списка и выводим на экран строку
    //с информацией по необходимому товару
    while(p!=NULL)
    {
        if (strcmp(p->fname,tempname)==0)
        {
            if (flag==0)
            {
                //выводим шапку таблицы
                printf("\n%s %s %s %s \n","# ","Merchant                    ","Amount","Price");
                printf("%s \n","-----------------------------------------");
                flag=1;
            }
            print(p);
        }
        p=p->pnext;
    }
    //не нашли товар, сообщаем об этом пользователю
    if (flag==0)
    {printf("%s %s\n","There are no merchant called - ",tempname);}
}
```

Опишем функции сортировки, которые запускается при вводе команд 8-10.

Команда 8 – функция сортировки включением:

```
//функция для сортировки включением
void insSort(Merchant *p, Merchant **begin, Merchant **end)
{

    Merchant *pnext1, *pnext2;
    //сдвигаем указатель на следующий элемент т.к. рассматривать будем всегда
    //текущий элемент и элемент левее его
    pnext1=p->pnext;
    while (pnext1!=NULL)
    {
        //запоминаем элемент следующий от двух рассматриваемых
        pnext2=pnext1->pnext;
        //пока рассматриваемый элемент не встанет на своё место в выравненном
        //в правильном порядке массиве, выполняем перстановку соседних элементов
        while (1)
            {
                //в случае если элемент стал первым в списке или если элемент
                //левее его меньше, заканчиваем делать перестановки
                if (pnext1->pprev==NULL) break;
                if (pnext1->price>pnext1->pprev->price) break;
                //корректируем указатель на начало или конец если необходимо
                if (pnext1->pprev->pprev==NULL) *begin=pnext1;
                if (pnext1->pnext==NULL) *end=pnext1->pprev;
                //сама функция перестановки
                pnext1=insInv(pnext1->pprev,pnext1);
            }
        //переставляем указатель на элемент следующий от двух рассматриваемых
        pnext1=pnext2;
    }
}
```

Вышеприведённая функция использует функцию перестановки элементов в списке:

```
//функция для смены местами элементов динамического массива
//работает как для смены соседних элементов так и для смены
//не соседних элементов
Merchant* insInv(Merchant *p1, Merchant *p2)
{
    Merchant *pnextTemp, *pprevTemp, *pprev2Temp, *pnextTemp2, *pprevTemp2;

    //здесь совершается смена указателей на объекты
    //таким образом делаем перестановку в двусвязном списке

    //при этом перестановка соседних элементов отличается от
    //перстановки не соседних, поэтому было разработано два блока

    if (p2->pprev==p1)//если элементы соседние
    {
    pprev2Temp=p1->pprev;
    pprevTemp=p1->pprev;
    pnextTemp=p2->pnext;

    p1->pnext=p2->pnext;
    p1->pprev=p2;

    p2->pnext=p1;
    p2->pprev=pprevTemp;

    if (pnextTemp!=NULL) pnextTemp->pprev=p1;
    if (pprev2Temp!=NULL) pprev2Temp->pnext=p2;
    }
    else//если элементы не соседние
    {
```

```
        pnextTemp=p2->pnext;
        pprevTemp=p1->pprev;
        pnextTemp2=p1->pnext;
        pprevTemp2=p2->pprev;

        p1->pnext=pnextTemp;
        p1->pprev=pprevTemp2;

        p2->pnext=pnextTemp2;
        p2->pprev=pprevTemp;

        pprevTemp2->pnext=p1;
        pnextTemp2->pprev=p2;
        if (pprevTemp!=NULL) pprevTemp->pnext=p2;
        if (pnextTemp!=NULL) pnextTemp->pprev=p1;
    }
    return p2;
}
```

Команда 9 – функция сортировки выбором:

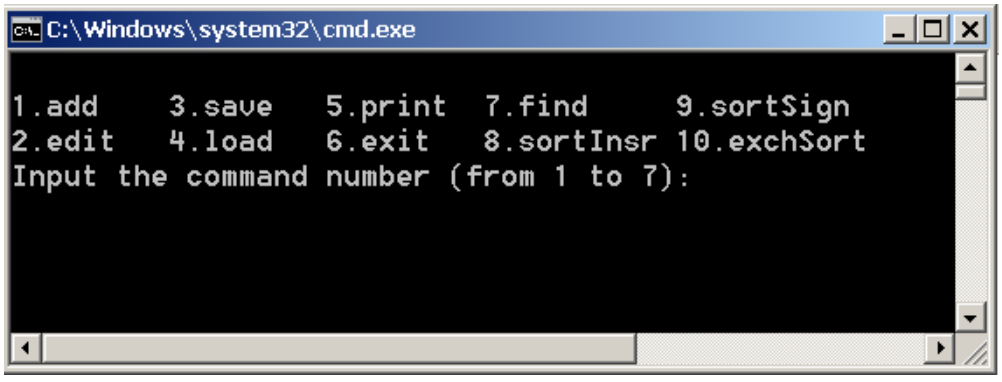
```
//функция для сортировки выбором
void signSort(Merchant *p, Merchant **begin, Merchant **end)
{
    int n=0,k=0,m=0,i;
    Merchant *pSign=p, *pCount=p, *pBegin=p;
    //считаем все элементы и находим наименьший
    while(pCount!=NULL){
        if (pCount->price<pSign->price) pSign=pCount;
        pCount=pCount->pnext;
        k++;}
    //если наименший не первый в очереди двигаем его на первое место
    //путём перестановки с первый элементом
    if (pSign!=p)
    {
        if (pSign->pnext==NULL) *end=p;
        pBegin=insInv(p,pSign);
        *begin=pBegin;//корректируем указатель на начало очереди
    }
    //данный счётчик отражает количество элементов которые стоят в необходимом порядке
    n++;
    //проходится по циклу будем (число элементов минус один) раз
    //с учётом уже выставленного в верном порядке одного элемента
    while(n<k-1)
    {
        pCount=pBegin; pSign=pBegin; m=0;
        //в данном цикле мы находим наименьшее значение ключа из оставшихся элементов
        while(pCount!=NULL)
        {
            if (m<n) pSign=pSign->pnext;
            if (pCount->price<pSign->price && m>=n) pSign=pCount;
            pCount=pCount->pnext;
            m++;
        }
        pCount=pBegin;
        //здесь мы находим первый элемент после выставленных в верном порядке эл-тов
        for(i=0;i<n;i++) pCount=pCount->pnext;
        //если первый элемент после выставленных в верном порядке не наименьший
        //меняем его местами с наименьшим
        if (pSign!=pCount)
        {
            if (pSign->pnext==NULL)
            {   //корректируем указатель на последний элемент если необходимо
                *end=pCount;
            }
            //сама функция перестановки элементов
            insInv(pCount,pSign);
        }
        n++;//увеличиваем число элементов выставленных в правильном порядке
    }
}
```

Команда 9 – функция сортировки обменом:

```
//функция для сортировки обменом
//принимает указатель на первый элемент очереди
//указатель на указатель на первый элемент очереди
//указатель на указатель на последний элемент очереди
void exchSort(Merchant *p, Merchant **begin, Merchant **end)
{

    int k=0;
    Merchant *pnext1=p, *pnext2=p, *pbegin=p;

    //считаем количество элементов в очереди и помещаем значение в k
    while(pnext1!=NULL)
    {
        k++;
        pnext1=pnext1->pnext;
    }
    //сравниваем по два элемента и переставляем их местами в случае выполнения условия
    //проходимся по всей очереди таким образом k-1 раз
    while (k-1>0)
    {
        //каждую итерацию цикла начинаем сравнивать со 2го элемента и сравниваем его с предыдущим
        pnext1=pbegin->pnext;
        while (pnext1!=NULL)
        {
            pnext2=pnext1->pnext;

            if (pnext1->price<pnext1->pprev->price)
            {
                //корректируем указатели на начало и конец очереди, в случае необходимости
                if (pnext1->pprev->pprev==NULL) {*begin=pnext1; pbegin=pnext1;}
                if (pnext1->pnext==NULL) *end=pnext1->pprev;
                //сама перестановка элементов
                pnext1=insInv(pnext1->pprev,pnext1);
            }
            //двигаемся на следующий элемент от двух рассматриваемых
            pnext1=pnext2;
        }
        //уменьшем количество необходимых итераций цикла прохода по очереди
        k--;
    }
}
```

Интерфейс работы программы:

Главное меню:



Печать списка товаров:

```
#  Merchant                  Amount Price
---------------------------------------
1  PC Acer                   5      1000
2  Mouse Cannon              1      10
3  Keyboard 57x              2      30
4  PC 2Core                  1      900
```

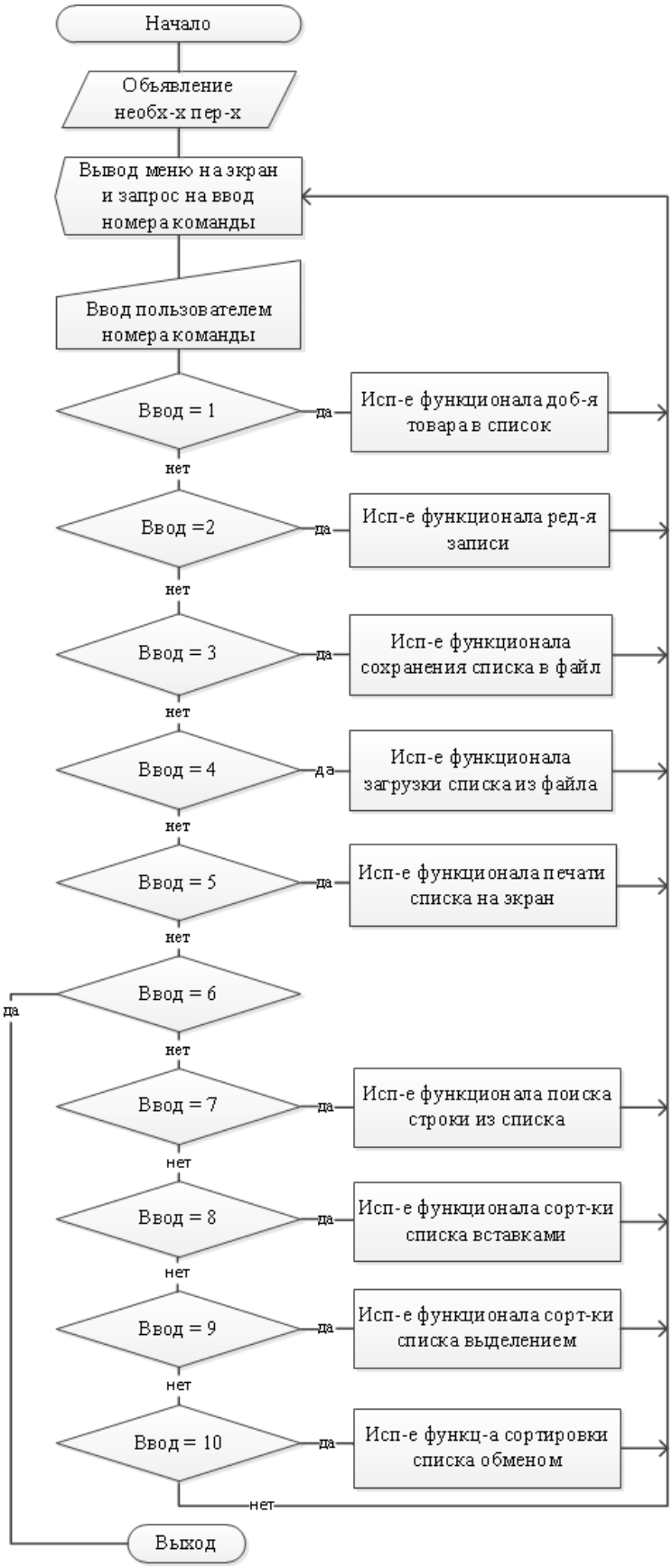
Редактирование записи о товаре:

```
1.add    3.save   5.print  7.find     9.sortSign
2.edit   4.load   6.exit   8.sortInsr 10.exchSort
Input the command number (from 1 to 7): 2

#  Merchant                  Amount Price
---------------------------------------
1  PC Acer                   5      1000
2  Mouse Cannon              1      10
3  Keyboard 57x              2      30
4  PC 2Core                  800    800
Input number of Merchant: 4

What do you want to do with it?
1.edit
2.delete
3.exit to the main menu
Input the command number : 1

Input merchant :PC 2Core
Input sold amount: 1
Input price: 900
```

Сортировка (по цене):

```
#  Merchant                  Amount Price
---------------------------------------
2  Mouse Cannon              1      10
3  Keyboard 57x              2      30
4  PC 2Core                  1      900
1  PC Acer                   5      1000
```

3. Блок-схема работы программы.



Заключение

В соответствии с индивидуальным заданием №1 разработана консольная программа учёта продаж программного обеспечения. Предоставляется возможность вести список программного обеспечения, и данных о его продажах: добавлять, редактировать удалять строки, сохранять информацию в файл и в последующем загружать её из файла. Программа прокомментирована, разобрана её структура. Отработаны навыки работы с массивами, структурами, строками, функциями, а также отработана запись информации в файл и её чтение из файла. Предусмотрен поиск информации по условию, вводимому пользователем с клавиатуры, а также сортировка по 3-м способам. Предоставлены ответы на теоретические вопросы.

Литература

1. Подбельский В.В., Фомин С.С. Программирование на языке Си: Учебное пособие. 2-е доп. изд. - М.: Финансы и статистика – 600 с.: ил.
2. Березин Б.И., Березин С.Б. Начальный курс С и С++. –М.: Диалог - МИФИ, 1999 - 288 с.
3. Основы алгоритмизации и программирования. Язык Си: учеб. пособие / М. П. Батура, В. Л. Бусько, А. Г. Корбит, Т. М. Кривоносова. - Минск: БГУИР, 2007. - 240 с.: ил.