ЗАТВЕРДЖЕНО

Наказ Міністерства освіти і науки,молоді та спорту України

29 березня 2012 року № 384

Форма № Н-6.01

**Державний вищий навчальний заклад «Донецький національний технічний університет»**

Кафедра комп’ютерної інженерії

КУРСОВИЙ ПРОЕКТ

(РОБОТА)

з програмування

на тему:

Розробка програмного забезпечення формування та обробки архіву адміністративно-облікових систем

Студентки 1 курсу групи КІ-13б

Напряму підготовки

«Комп’ютерна інженерія»

спеціальності «Комп’ютерні системи»

Ахмедової Д.Н.

Керівник Юсупова К.Б.

Національна шкала \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_\_\_\_\_\_Оцінка: ECTS

Члени комісії

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (прізвище та ініціали)

м. Донецьк — 2014 рік

Затверджую

Зав. кафеди КІ\_\_\_\_\_\_\_

Святний В. А.

\_\_\_\_\_\_\_ 2012 року.

# Техническое задание к курсовой работе по дисциплине «программирование»

Тема роботи: «Розробка програмного забезпечення формування та обробки архіву адміністративно-облікових систем».

Студентки групи КІ-13б Ахмедової Дарина.

Термін здачі роботи на перевірку: згідно з графіком виконання.

Варіант № 21.

* У книзі заявок житлового ремонтного управління містяться відомості про заявки: прізвище та ініціали заявника, його адреса, тип ремонту (малий, середній, великий), дати заявки і планового терміну виконання ремонту, фактичний термін виконання ремонту (початок і кінець). У планових і фактичних термінах вказувати рік, місяць, день. Надрукувати відомості про заявників, у яких ремонт виконаний із запізненням в три і більше місяців, а також відомості про три першочерговиків на малий ремонт. Коректовані реквізити: дати ;
* Тип лінійного списку: черга.
* Метод сортування: прямого вибору.
* Опис функції *AddArchive.c*

Мета — освоїти методи проектування багатомодульних програм, у яких обробляються лінійні списки, інформаційної частиною який є структури, для реалізації функцій адміністративно-облікової системи (АОС).

## Вимоги до виконання:

1. Програмне забезпечення повинно бути реалізовано як консольний додаток WIN-32, мова програмування — мова класу С, інструментальне оточення — Microsoft Visual C.
2. Введення записів з текстового файлу і формування типізований файлу, який розглядається надалі як архів АОС.
3. Введення записів кодификатора з текстового файлу і формування масиву, який має бути відсортованим за певною ознакою.
4. Групування архіву за певною ознакою.
5. Друк на екрані і запис в текстовий файл архіву АОС і кодификатора.
6. Доповнення архіву новими записами.
7. Видалення заданої записи з архіву.
8. Редагування реквізитів заданого запису.
9. Формування вихідних документів.
10. Контроль текстових файлів на помилки формату, помилки діапазону і змістовні помилки.
11. Вхідні текстові файли архіву та кодификатора повинні містити не менше 20 записів. Доповнення архіву має проводитися з використанням окремого текстового файлу, який містить 3-5 записів.

## Зміст пояснювальної записки:

Технічне завдання

Реферат

Зміст

Вступ

1. Структура вхідних даних.
2. Структура програмного забезпечення (ПЗ) і головної програми.
3. Опис ПЗ.
   1. Побудова динамічного списку.
   2. Групування архіву певним методом.
   3. Алгоритм обробки динамічного списку: додавання даних у архів.
   4. Обробка архіву.
   5. Організація роботи з кодифікатором.
4. Тестування ПЗ і дослідження результатів.
5. Технічні характеристики програмних модулів.

Висновок

Список використаної літератури.

Додаток А — Приклади архіву та кодифікатора.

Додаток В — Лістинги програм

## Графік виконання:

1. Аналіз завдання. Розробка структури та змісту архіву та кодифікатора (1-2 тиждень).
2. Контроль інформації в текстових файлах (3-4 тиждень).
3. Створення архіву в бінарному файлі і друк архіву у вигляді таблиці на екран і запис у файл. Створення та сортування масиву кодифікатора (5-6 тиждень).
4. Сортування архіву заданих методом (7 тиждень).
5. Розробка алгоритмів обробки архіву: додавання, видалення і редагування записів архіву (8-9 тиждень).
6. Формування звітів згідно із завданням (10-11 тиждень).
7. Розробка багатомодульної програми. Тестування ПЗ (12-13 тиждень).
8. Оформлення пояснювальної записки. Захист курсової роботи (14-15 тиждень).

Керівник роботи \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Завдання прийняв до виконання \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Реферат Пояснювальна записка до курсової роботи: стор., табл., рис., додатків, посилань  Об’єктом розробок та досліджень є адміністративно-облікова система.  Ціль роботи — освоїти методи проектування багатомодульних програм, в яких обробляються лінійні списки, інформаційної частиною яких є структури, для реалізації функцій адміністративно-облікової системи (АУС).  Програму написано з використанням мови програмування Microsoft Visual C. Програма скомпільована для платформи Windows XP, Vista, 7, 8, 8.1.  В результаті роботи було розроблено програмне забезпечення, що призначено для перегляду, редагування та фільтрації адміністративно-облікової системи.  ДИНАМІЧНИЙ СПИСОК, ЧЕРГА, КОДІФІКАТОР, БАГАТОМОДУЛЬНА ПРОГРАМА | | | | | | | | | | |
|  |  |  |  |  | КР 7.091501-08-29532.00.00.000 ПЗ | | | | | |
|  |  |  |  |  |
| Вим | Лист | № докум | Підп | Дата |
| Розр. | | Ахмедова Д.Н. |  |  | Розробка програмного забезпечення формування та обробки архіву адміністративно-облікових систем | Лит | | | Лист | Листів |
| Керів. | | Юсупова К.Б. |  |  |  |  |  |  |  |
| Т.контр. | | Дорожко Л.І. |  |  | ДонНТУ, кафедра КІ  Гр. КІ-13б, | | | | |
| Н.контр. | | Череднікова О.Ю. |  |  |
| Зав.каф. | | Святний В. А. |  |  |

# Зміст

[Вступ](#_Toc389423042) 7

[1 Структура вхідних даних 7](#_Toc389423043)

[2 Структура ПЗ та головної програми 9](#_Toc389423044)

[3 Опис ПЗ 12](#_Toc389423045)

[3.1 Побудова дінамічного списку 12](#_Toc389423046)

[3.2 Групування архіву визначеним методом 13](#_Toc389423047)

[3.3 Додавання даних у архів 14](#_Toc389423048)

[3.4 Видалення даних з архіву 1](#_Toc389423049)6

[3.5 Обробка архіву 17](#_Toc389423050)

[3.6 Організація роботи з кодіфікатором 1](#_Toc389423051)8

[3.7 Зміна компонентів в архіві](#_Toc389423051) 20

[4 Тестування ПЗ](#_Toc389423052) 23

[5Технічні характеристики](#_Toc389423052) 33

[Висновок](#_Toc389423052) 34

[Список використано\ літератури](#_Toc389423053) 35

# Вступ

Метою даної курсової роботи є розробка адміністративно-облікової системи (АОС) для керування записами бази даних.

Розроблена програма повинна дозволяти:

* вводити записи з текстового файлу і формувати типізований файл (архів АОС);
* вводити записи кодифікатора з текстового файлу і формувати масив кодифікатора;
* групувати архів за певною ознакою;
* друкувати на екрані і записувати в текстовий файл архів АОС і кодифікатор;
* доповнювати архів новими записами;
* видаляти заданий запис з архіву;
* редагувати реквізити заданого запису;
* формувати вихідні документи;
* контролювати текстові файли на помилки формату, помилки діапазону і змістовні помилки.

Програма повинна бути написана з використанням мови програмування C в інструментальному оточенні Microsoft Visual C.

# 1 Структура вхідних даних

Згідно з технічним завданням, вхідні дані мають вигляд:

Таблиця 1 — формат загальних даних

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Код** | **П.І.Б.** | **Вулиця** | **Тип ремонту** | **Плановий термін виконання роботи** | **Фактичний термін виконання роботи** |
| 12 | Шуригіна\_Т\_Ю | Постишева | м | 20140317- 20140320 | 20140317- 20140517 |
| 63 | Любченко\_Н\_А | Байрона | б | 20130510- 20140820 | 20130510- 20131023 |
| *цілий* | *рядок* | *рядок* | *рядок* | *цілий* | *цілий* |

Поля «Код», «Плановий термін виконання роботи», «Фактичний термін виконання роботи» представляють собою цілочисельні дані, тому їх переважно зберігати в змінній типу int. Поля «Прізвище Ім’я По батькові», «Вулиця», «Тип ремонту» представляють масив букв, їх треба зберігати в строковому вигляді.

Кожен запис основних даних зберігається в структурі:

/\*

typedef struct предназначен для хранения информации.

\*/

typedef struct {

int Kod ;

char Family[15] ; // ФИО

char Adress[15];

char Remont[10];

int Data1;

int Data2;

int Data3;

int Data4;

} ProductType;

Рисунок 1.1 Структура вхідних даних

Поле «№», «Код», «Номер телефону» представляють цілочисельні дані, тому його переважно зберігати в змінній типу int.

Таблиця 2 — формат даних кодифікатора

|  |  |  |
| --- | --- | --- |
| **№** | **Код** | **Номер телефону** |
| 1 | 12 | 0665125464 |
| 2 | 14 | 0992354678 |
| *цілий* | *цілий* | *цілий* |

Усі вхідні дані (дані основного формату і кодифікатор) зберігаються в файлах типу .txt. Файл архіву є бінарним файлом, що містить послідовність записів.

typedef struct { //тип компонента кодификатора изделий

int Kod; // код изделия

char Name[35]; // наименование изделия

} KodifType;

Рисунок 1.2 Структура вхідних даних

# 2 Структура ПЗ та головної програми

Створювана програма здатна працювати в 9 режимах:

1. Контроль текстових файлів - функція checkFiles(), файли CheckInpFile.c CheckInpFile.h  
   2. Створення архівy - функція createArchive(), файл CreateArchKodif.c CreateArchKodif.h  
   3. Сортування компонент архіву - функція SortArchive(), файл SortArchiv.c SortArchiv.h  
   4. Друк архіву виробів - функція PrintArchive(), файл CreateArchKodif.c CreateArchKodif.h  
   5. Друк кодифікатора виробів - функція PrintKodif(), файл CreateArchKodif.c CreateArchKodif.h  
   6. Додавання компонент в архів - функція AddArchive(), файл AddArchiv.c AddArchiv.h  
   7. Видалення компоненти з архіву - функція DeleteArchive(), файл DeleteArchiv.c DeleteArchiv.h  
   8. Зміна компоненти в архіві - функція ChangeArchive(), файл EditArchiv.c EditArchiv.h  
   9. Обробка архіву - функція WorkUpArchive (), файл ReportArchiv.c ReportArchiv.h
2. Загальна структура програми:
3. Модуль *CheckInpFile.h/.c* містить функції, пов'язані з перевіркою форматів файів: перевірка наявності і введення текстового файлу, контроль кількості рядків, і видалення в ньому порожніх рядків, перевірка форматів файлу "Kodif.txt", перевірка параметрів записів у файлах "input.txt" і "add.txt", перевірка дублювання параметра KodProduct у файлі "Kodif.txt", читання і виведення на екран протоколу контролю текстових файлів.
4. Модуль *CreateArchKodif.h/.c* містить такі функції: зчитується рядок з вказаного файлу, проводиться її розбір і формування структури Produkt, зчитуються рядки опису продукту з текстового файлу формується бінарний файл архіву. З текстового файлу зчитується кодифікатор і записується в масив. Масив сортується і зберігається до закриття програми. З бінарного файлу зчитується архів і друкується на екрані, в файл.Чтеніе файлу "Kodif.txt" і формування масиву Kodifs. Висновок кодификатора виробів на екран, у файл
5. Модуль *SortArchive.h/.c* містить визначення стека і функції, пов’язані з його обробкою. Сортування архіву по зростанню коду виробу. Перегляд черги і сортування за спаданням "шкільним" методом. Читання з бінарного файлу архіву і формування черги записів і включення елементів в чергу..
6. Модуль *AddArchive.h/.c* містить визначення стека і функції, пов’язані з його обробкою та додає компоненти в архів.
7. Модуль *DeleteArchive.h* містить визначення стека і функції, пов’язані з його обробкою та видаляє компоненти з архіву.
8. Модуль *EditArchve.h/.c* містить визначення стека і функції, пов’язані з його обробкою та зміна компонентів в архіві.
9. Модуль *Buseunit.h/.c* Висновок повідомлення на екран і очікування натискання клавіші. Висновок рядка на екран, і в файл. Пошук в рядку пробілу та символу відмінного від пробілу. Введення числа з клавіатури. Пошук Коду та сортування його «шкільним» методом
10. Модуль *ReportArchive.h/.c* містить визначення стека і функції, пов’язані з його обробкою: обробка архіву виробів і формування звітів.
11. Модуль *Desunit.h*  - описані всі структури.
12. Модуль *Kurs.h/.c* виклик функцій:

extern void CreateArchive();

extern int SortArchive();

extern int PrintArchive();

extern int PrintKodif(int nk);

extern void CheckFiles();

extern int AddArchive();

extern int DeleteArchive();

extern int ChangeArchive();

extern void WorkUpArchive();.

Рисунок 1.3 – функції

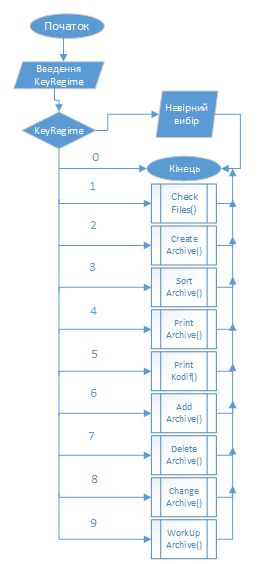


Рисунок 1.4 — Загальна структура ПЗ

# 3 Опис ПЗ

## 3.1 Побудова дінамічного списку

У програмі, що створюється, використовується динамічний список-черга. Визначення черги та функцій для роботи з нею знаходяться у модулі *SortArchiv.h/.c*.

Графічно чергу можна представити у виді, який показано нижче:

inf

inf

next

next

inf

next

inf

next

NULL

L

R

Рисунок 3.1 — Побудова черги

де *L* — вказівник на лівий елемент черги, *R* — вказівник на правий елемент черги*, inf* — поле, що містить інформацію, а *next* — вказівник на наступний елемент черги.

У програмі черга зберігається у структурі:

int ReadFileOut(int \*np, DynProduct \*\*L, DynProduct \*\*R);

int WriteFileOut(DynProduct \*L);

void DisposeProduct(DynProduct \*L);

Функція *AddArchiv()* призначена для додавання елементу в чергу.

Функція *DeleteArchiv()* видаляє всі елементи черги

Функція *SortArchiv()* виконує реверс черги.

Функція WorkUpArchiv*()* обробка черги по заданих параметрах.

## 3.2 Групування архіву визначеним методом

У даній курсовій роботі реалізовано один метод сортування архіву: сортування «шкільним» методом.

Метод реалізован у модулі *SortArchv.h/.c*.

Прототипи функцій, що виконують сортування. Її прототип ми бачимо нижче.

int SortArchive();

int ReadFileOut(int \*np, DynProduct \*\*L, DynProduct \*\*R);

int WriteFileOut(DynProduct \*L);

void DisposeProduct(DynProduct \*L);

Рисунок 3.2— Прототипи функцій, що виконують сортування

Функція SortArchive() виконує сортування «шкільним» методом.



Рисунок 3.3 — Сортування черги*.*

3.3 Додавання даних у архів

У курсовій передбачена функція додавання даних до архіву із зовнішнього файлу. Вважається, що файл містить коректні дані.

Ця функція визначена у модулі *AddArchiv.h/.c*; її прототип:

int AddArchive();

extern void wait\_press\_key(char \*msg);

//SortArchive

extern int ReadFileOut(int \*np, DynProduct \*\*L, DynProduct \*\*R);

extern int WriteFileOut(DynProduct \*L);

extern void DisposeProduct(DynProduct \*L);

//CreateArchKodif

extern int ReadProduct(FILE \*f, ProductType \*Product);

Рисунок 3.4— прототип

Функція ReadFileOut( ) читання з бінарного файлу архіву, формування черги записів та включення елементів в чергу(включаємо нові єлементи з кінця). Функція WriteFileOut () пересилка записів з черги в бінарний файл архіву. Перегляд з правого боку. Далі видаляємо чергу. Функція DisposeProduct *()* видалення черги з початку списку.

При додаванні елемента в чергу модифікується вказівник на хвіст черги і записується новий елемент.

Вставляє новий елемент у відносну по-зицію, що задається параметром УКАЗАТЕЛЬ. Якщо в цій позиції вже є елемент, то він "про-талківается вниз", щоб звільнити местодля нового елемента. Усі наступні покажчики перебудовуються з урахуванням нового елемента. Наприклад, елемент, доданий в позіцію10, виштовхне старий десятий елемент в позицію 11, еелмент 11 - в плзіцію 12 і т.д. Як що ПОКАЖЧИК = 0 або більше загальної кількості елементів ЧЕРЗІ, то елемент буде додано в кінець черги.



Рисунок 3.5 – добавление в архив

3.4 Видалення даних з архіву

У курсовій передбачена функція додавання даних до архіву із зовнішнього файлу. Вважається, що файл містить коректні дані.

Ця функція визначена у модулі Delete*Archiv.h/.c*;

//baseunit

extern void wait\_press\_key(char \*msg);

extern double GetNumber(double MinNumber, double MaxNumber,

char m1,char n1,char m2,char n2);

//SortArchive

extern int ReadFileOut(int \*np, DynProduct \*\*Beg);

extern int WriteFileOut(DynProduct \*Beg);

extern void DisposeProduct(DynProduct \*Beg);

//--------------------------------------------------------

int DeleteArchive();

Рисунок 3.6 — прототип

Функція ReadFileOut( ) читання з бінарного файлу архіву, формування черги записів та включення елементів в чергу(включаємо нові єлементи з кінця). Функція WriteFileOut () пересилка записів з черги в бінарний файл архіву. Перегляд з правого боку. Далі видаляємо чергу. Функція DisposeProduct *()* видалення черги з початку списку.

При видаленні вибирається елемента, на який вказує покажчик на голову черги, після чого цей покажчик модифікується.

1. Запам'ятовуємо вказівник на голову списку
2. Перевизначають вказівник на наступний після нього елемент
3. Звільняємо пам'ять, яку запам'ятали в пункті 1

Також нижче представлена блок-схема видалення даних з архіву.

Також нижче представлена блок-схема видалення даних з архіву.



Рисунок 3.7 — видалення даних з архіву

## 3.5 Обробка архіву

Програма, зо реалізується, повинна мати режим обробки архіву за наступним завданням:

«У книзі заявок житлового ремонтного управління надрукувати відомості про заявників, у яких ремонт виконаний із запізненням в три і більше місяців, а також відомості про три першочерговиків на малий ремонт.»

Обробка архіву реалізована у функції WorkUpArchive *()*, що міститься у модулі *ReportArchiv.h/.c*.





Рисунок 3.8 — обробка архіву

## 3.6 Організація роботи з кодіфікатором

У даній курсовій роботі використовується кодифікатор. Кодифікатор — це додатковий файл пар виду «Ключ — значення», де ключ зазвичай є цілим числом, а значення — рядком. Його призначення — зменшення розмірів файлів головного формату. Наприклад, у ситуації, коли якісь з полів записів можуть часто дублюватися, їх значення можна замінити ключом кодифікатора. У цій курсовій ключами кодифікатора замінюються номера телефонів людей, котрі подали заявки на житловий ремонт .

У програмі кодифікатор та всі функції для роботи з ним зберігаються у модулі *CreateArchKodif.h/.c*. У цьому файлі визначені змінні:

extern KodifType Kodif, //компонент кодификатора

Kodifs[50]; //массив компонентов кодификатора

extern int nk; // кол-во компонентов кодификатора

Рисунок 3.9—Змінні модуля *CreateArchKodif.h/.c*.

та функції:

extern void SortKodif(int nk);

int PrintKodif(int nk);

Рисунок 3.10 — прототипи функції модуля *CreateArchKodif.h/.c*.

Функція SortKodif*()* виконує ­­­­сортування кодифікатора.

Функція PrintKodif *()* друкує кодифікатор на екран.

# 3.7 Зміна компонентів в архіві

Програма, зо реалізується, повинна мати режим зміни компонентів в архіві за наступним завданням:

«Коректовані реквізити: номер цеху , табельний номер і розряд ;»

Зміна компонентів в архіві реалізована у функції ChangeArchive *()*, що міститься у модулі *EditArchiv.h/.c*. Протопип цієї функції:

//baseunit

extern void wait\_press\_key(char \*msg);

extern int Space(char \*s, int k);

extern int NotSpace(char \*s, int k);

extern char \*FillString(char \*S, unsigned char len, unsigned char pk);

extern double GetNumber(double MinNumber, double MaxNumber,

char m1,char n1,char m2,char n2);

//SortArchive

extern int ReadFileOut(int \*np, DynProduct \*\*L, DynProduct \*\*R);

extern int WriteFileOut(DynProduct \*L);

extern void DisposeProduct(DynProduct \*L);

//------------------------------------------------------

int ChangeArchive();

void MakeComponent(ProductType \*Product);

Рисунок 3.11 — прототип



Рисунок 3.12 — зміна компонентів архіву

# 4 Тестування ПЗ

При запуску програми з’являється меню програми (рис. 4.1).

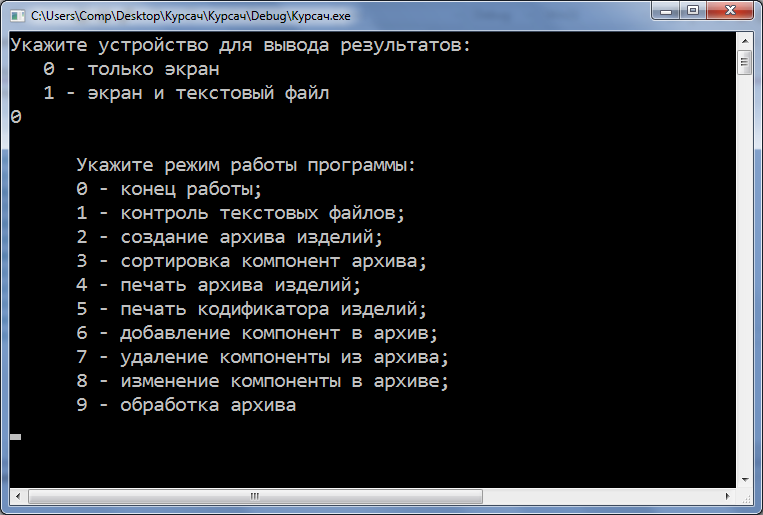


Рисунок 4.1 — Меню програми

Перший режим роботи — це контроль текстових файлів (рис. 4.2). Якщо файли некоректні, з’являється опис помилки (рис. 4.3).

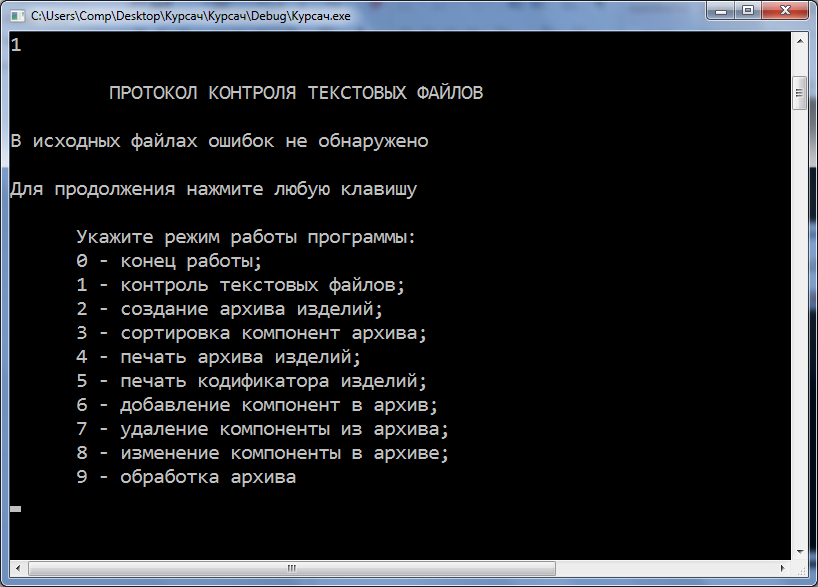


Рисунок 4.2 — Перший режим: файли коректні

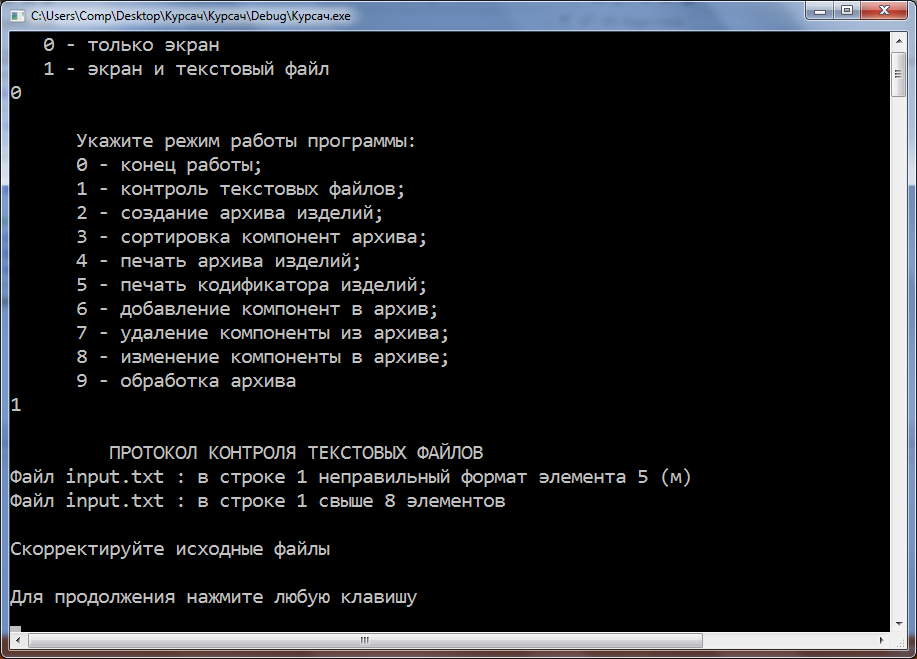


Рисунок 4.3 — Перший режим: файли некоректні

Другий режим — це створення архіву(рис. 4.4).

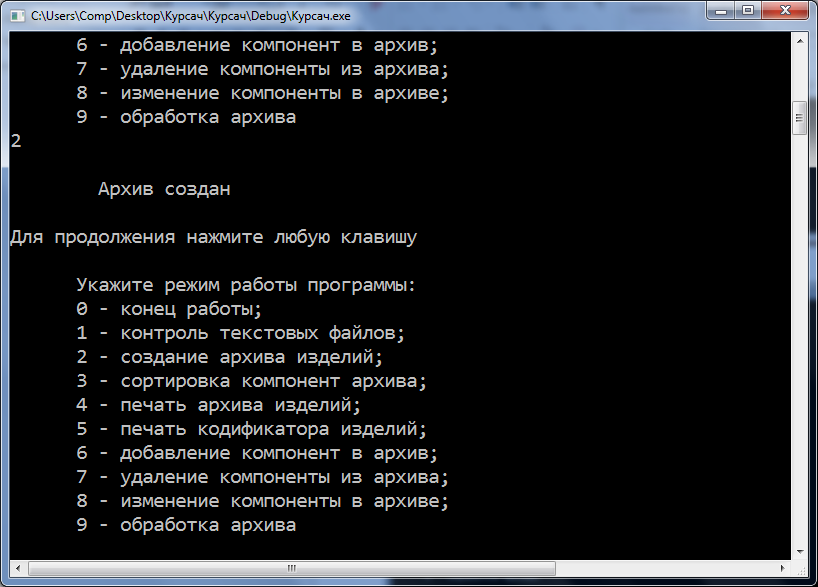


Рисунок 4.4 — Другий режим

Четвертий режим — це друк архіву (рис. 4.5).

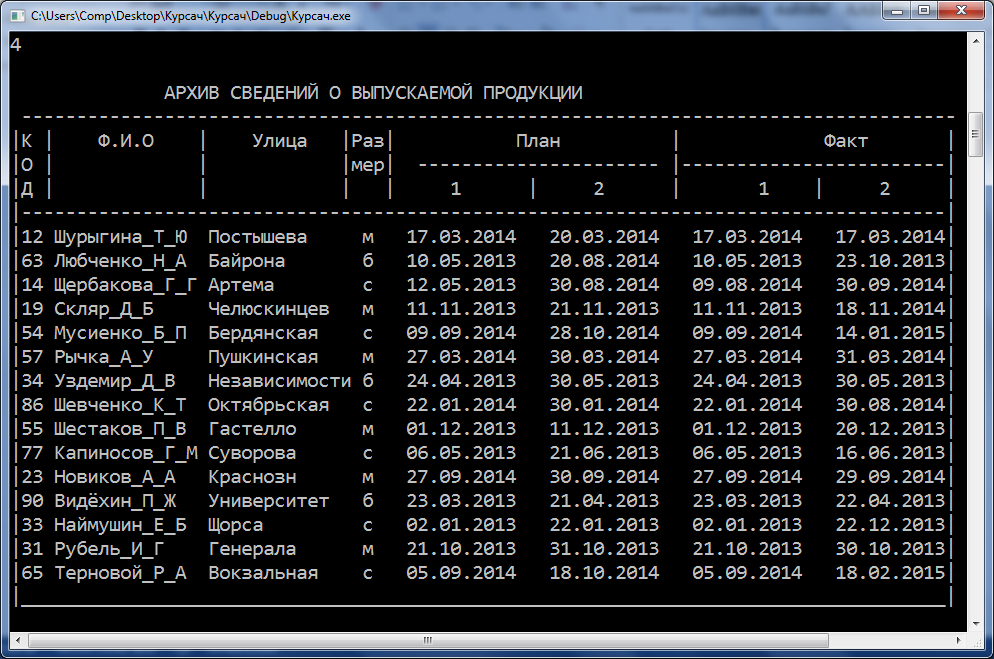


Рисунок 4.5 — Четвертий режим

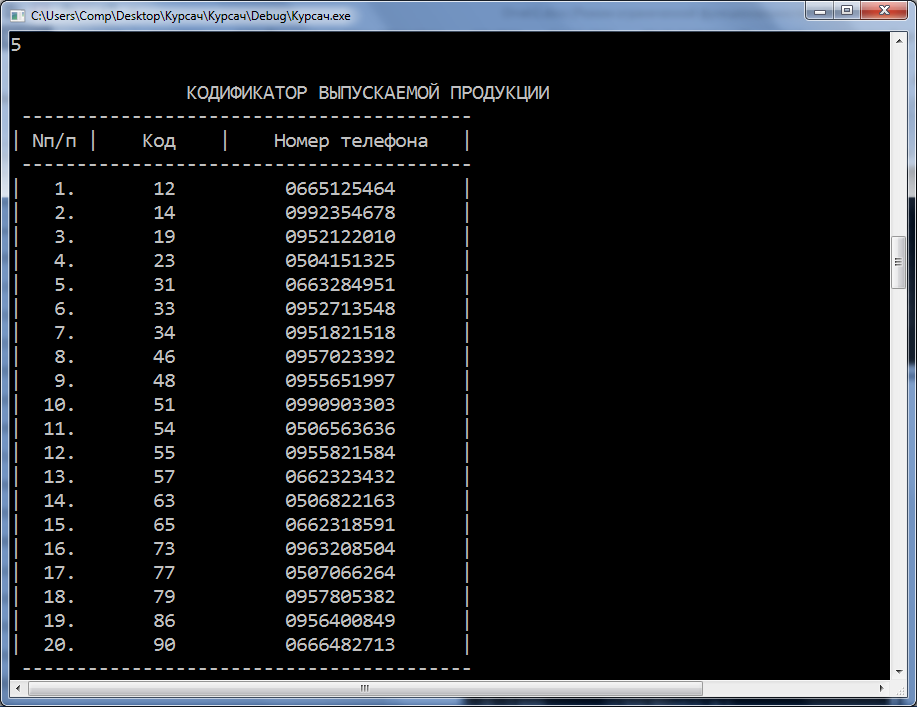
П’ятий режим — це друк кодифікатору (рис. 4.6).

Рисунок 4.6 — П’ятий режим

Третій режим — це сортування архіву (рис. 4.7).

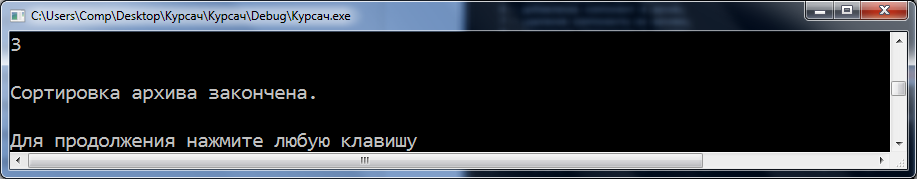
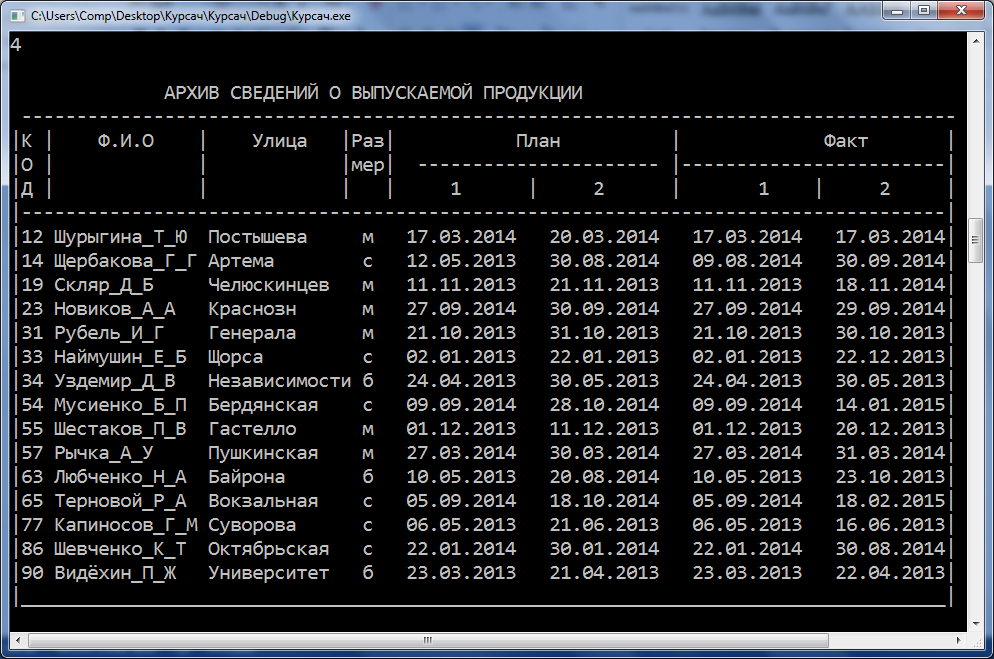
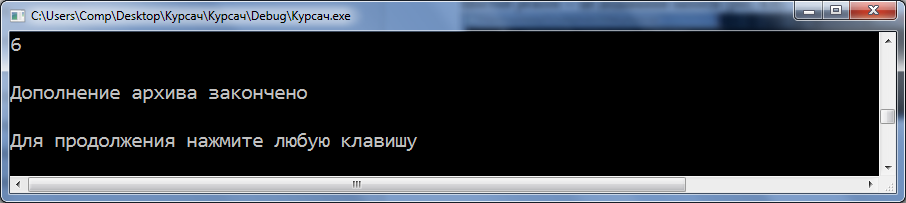


Рисунок 4.7 — Третій режим

Шостий режим — це додавання записів (рис. 4.8).



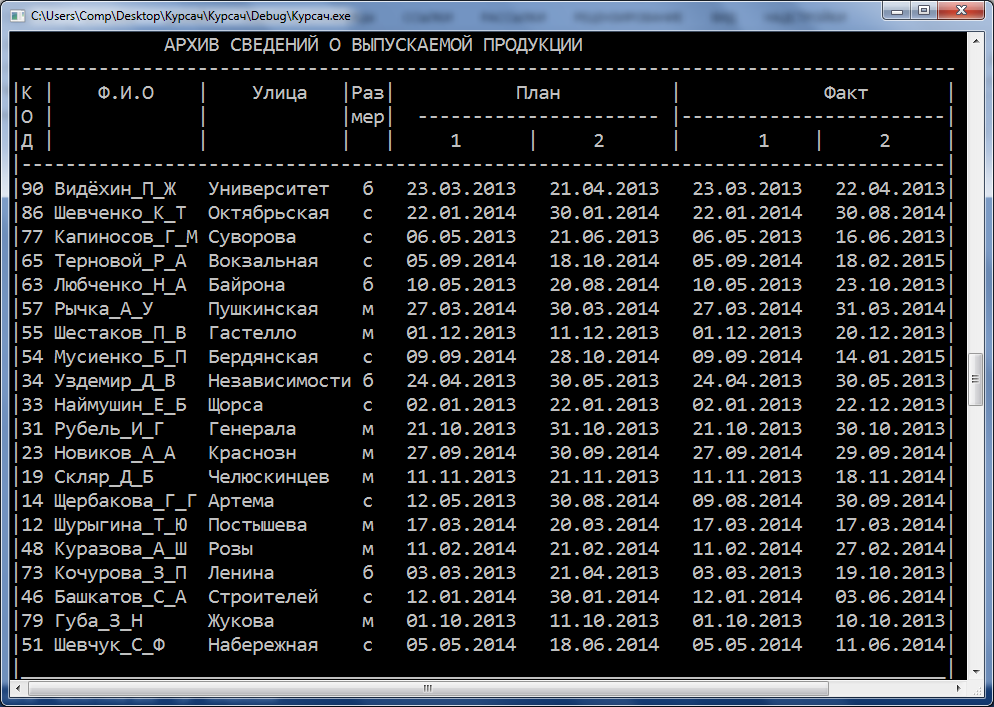
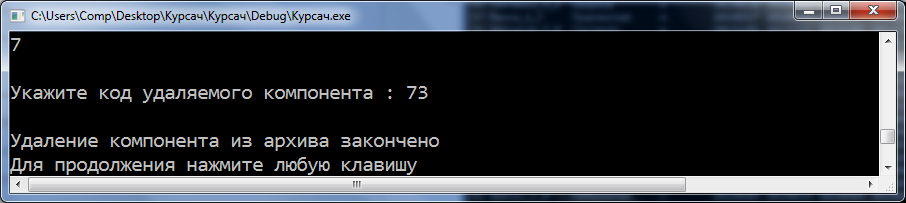


Рисунок 4.8 — Шостий режим

Сьомий режим — це видалення запису з архіву (рис. 4.9).



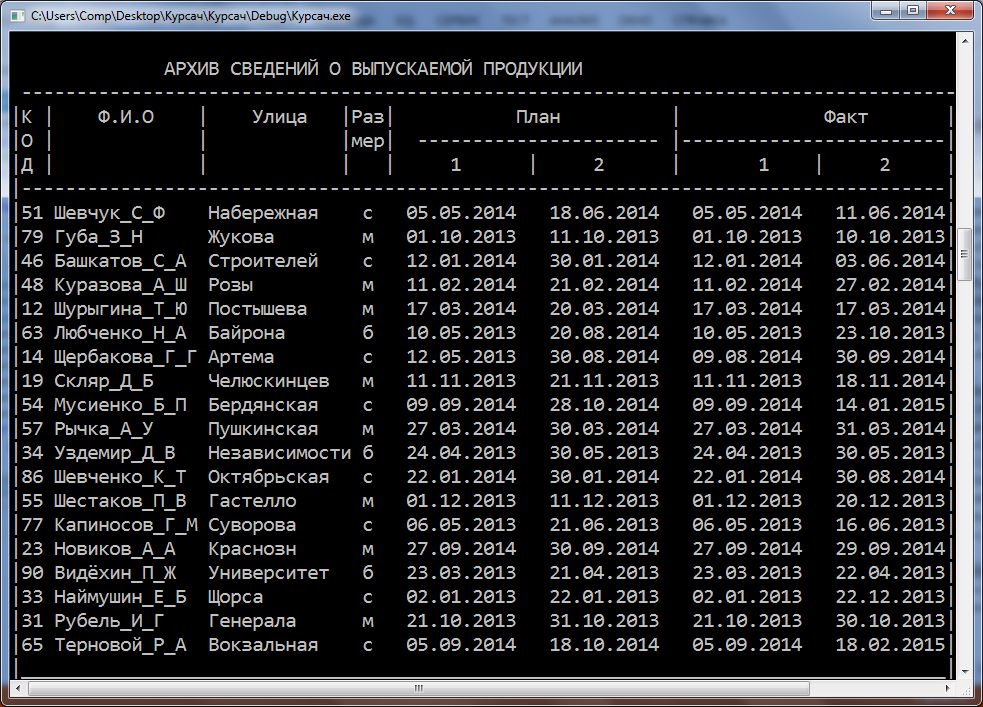
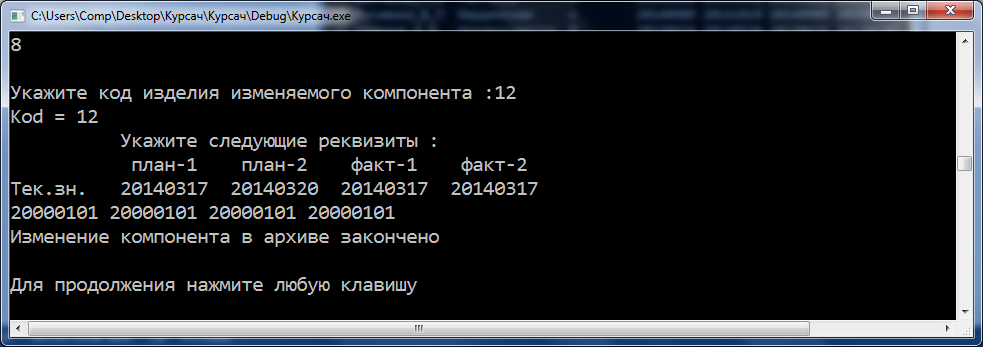


Рисунок 4.9 — Сьомий режим

Восьмий режим — це зміна запису в архіві (рис. 4.10).



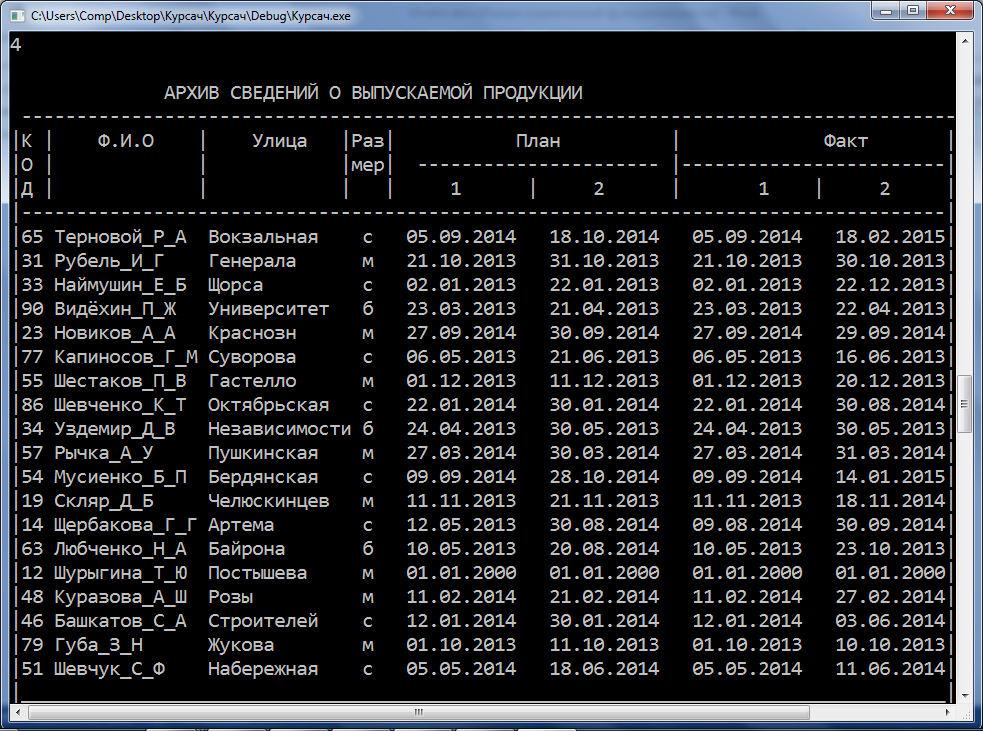


Рисунок 4.10 — Восьмий режим

Дев’ятий режим — це виконання головної задачі (рис. 4.11).

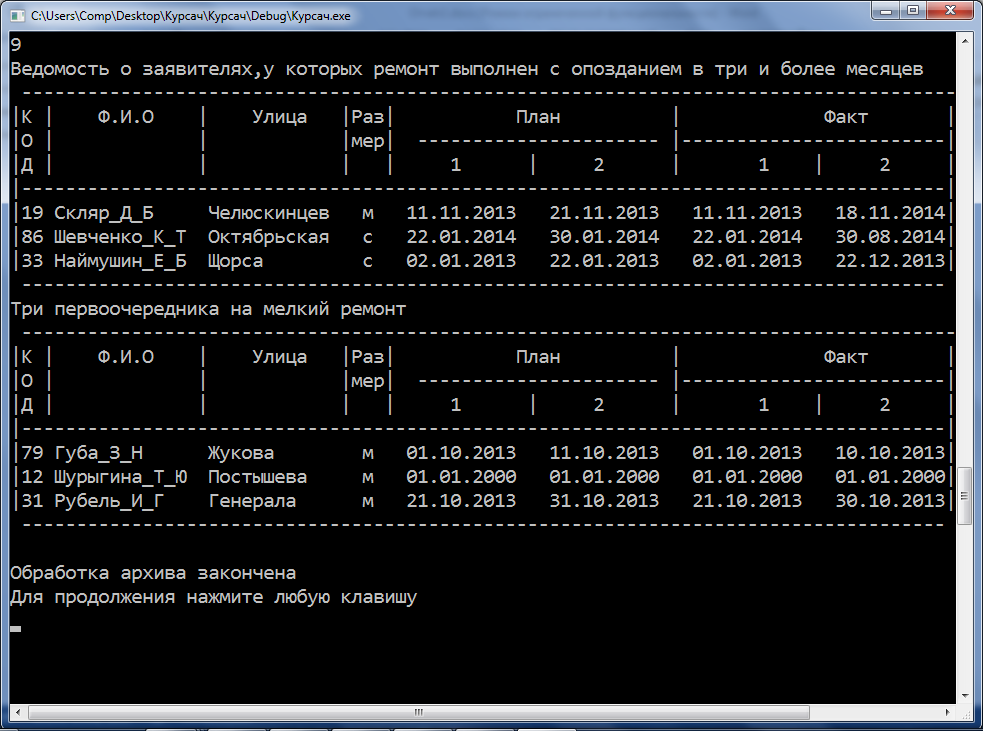


Рисунок 4.11 — Дев’ятий режим

5 Технічні характеристики програмних модулів

Мінімальними технічними характеристиками для роботи програми є:

* ОС: Windows XP або новіше
* Оперативна пам’ять: від 128 Мб
* Процесор: Intel Pentium 4 або новіше

Обсяг вихідного коду сягає 45.4 Кб

# Висновок

Таким чином, розроблена програма дозволяє:

* вводити записи з текстового файлу і формувати типізований файл (архів АОС);
* вводити записи кодифікатора з текстового файлу і формувати масив кодифікатора;
* групувати архів за певною ознакою;
* друкувати на екрані і записувати в текстовий файл архів АОС і кодифікатор;
* доповнювати архів новими записами;
* видаляти заданий запис з архіву;
* редагувати реквізити заданого запису;
* формувати вихідні документи;
* контролювати текстові файли на помилки формату, помилки діапазону і змістовні помилки.

Програма повинна написана з використанням мови програмування C в інструментальному оточенні Microsoft Visual C.

# Список використаної літератури

1. Хомоненко А.Д. Основы современных компьютерных технологий//Учебное пособие для вузов. – Ст-Петербург: Корона принт, 1998.
2. Википедия – свободная энциклопедия http://ru.wikipedia.org/.
3. *Брайан Керниган, Деннис Ритчи.* Язык программирования Си. — Санкт-Петербург: Невский диалект, 2001. — 352 с. — (Библиотека программиста). — [ISBN 5794000457](http://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D1%83%D0%B6%D0%B5%D0%B1%D0%BD%D0%B0%D1%8F:%D0%98%D1%81%D1%82%D0%BE%D1%87%D0%BD%D0%B8%D0%BA%D0%B8_%D0%BA%D0%BD%D0%B8%D0%B3/5794000457)
4. К. Поляков. Прграммирование на языке СИ. 2002 год. 4 главы в отдельных файлах PDF. Размер архива 2.7 Мб.
5. Шпак З.Я. – Програмування мовою С: Навчальний посібник

# Додаток А — Приклади архіву та кодифікатора

Приклад архіву зображений на рис. А.1.

Рисунок А.1 — Приклад архіву

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 12 | Шурыгина\_Т\_Ю | Постышева | м | 20140317 | 20140320 | 20140317 | 20140317 |
| 63 | Любченко\_Н\_А | Байрона | б | 20130510 | 20140820 | 20130510 | 20131023 |
| 14 | Щербакова\_Г\_Г | Артема | с | 20130512 | 20140830 | 20140809 | 20140930 |
| 19 | Скляр\_Д\_Б | Челюскинцев | м | 20131111 | 20131121 | 20131111 | 20131118 |
| 54 | Мусиенко\_Б\_П | Бердянская | с | 20140909 | 20141028 | 20140909 | 20150114 |
| 57 | Рычка\_А\_У | Пушкинская | м | 20140327 | 20140330 | 20140327 | 20140331 |
| 34 | Уздемир\_Д\_В | Независимости | б | 20130424 | 20130530 | 20130424 | 20130530 |
| 86 | Шевченко\_К\_Т | Октябрьская | с | 20140122 | 20140230 | 20140122 | 20140830 |
| 55 | Шестаков\_П\_В | Гастелло | м | 20131201 | 20131211 | 20131201 | 20131220 |
| 77 | Капиносов\_Г\_М | Суворова | с | 20130506 | 20130621 | 20130506 | 20130616 |
| 23 | Новиков\_А\_А | Краснознаменная | м | 20140927 | 20140930 | 20140927 | 20140929 |
| 90 | Видёхин\_П\_Ж | Университетская | б | 20130323 | 20130421 | 20130323 | 20130422 |
| 33 | Наймушин\_Е\_Б | Щорса | с | 20130102 | 20130122 | 20130102 | 20130222 |
| 31 | Рубель\_И\_Г | Генерала | м | 20131021 | 20131031 | 20131021 | 20131030 |
| 65 | Терновой\_Р\_А | Вокзальная | с | 20140905 | 20141018 | 20140905 | 20150218 |

Приклад кодифікатора зображений на рис. А.2.

12 0665125464

63 0506822163

14 0992354678

19 0952122010

54 0506563636

57 0662323432

34 0951821518

86 0956400849

55 0955821584

77 0507066264

23 0504151325

90 0666482713

33 0952713548

31 0663284951

65 0662318591

48 0955651997

73 0963208504

46 0957023392

79 0957805382

51 0990903303

Рисунок А.2 — Приклад кодифікатора

# Додаток В — Лістинги програм

DESUNIT.h

#include<stdio.h>

#include<math.h>

#include<string.h>

#include<conio.h>

#include<windows.h>

#define fArTxtName "input.txt"

#define fArBinName "arhiv.bin"

#define fAddTxtName "add.txt"

#define fResName "res.txt"

#define fKodifName "kodif.txt"

typedef struct {

int Kod ;

char Family[15] ; // ФИО

char Adress[15];

char Remont[1];

int Data1;

int Data2;

int Data3;

int Data4;

} ProductType;

typedef struct { //тип компонента кодификатора изделий

int Kod; // код изделия

char Name[15];

} KodifType;// наименование изделия

typedef struct DynProd {

ProductType Inf;

struct DynProd \*Next;

} DynProduct;

BASEUNIT.H

#include "desunit.h"

//-----------------------------------------------------------

FILE \*fArTxt;

FILE \*fArBin;

FILE \*fAddTxt;

FILE \*fKodif;

KodifType Kodif, //компонент кодификатора

Kodifs[50]; //массив компонентов кодификатора

DynProduct \*PointerProduct ;

unsigned char SignArchive=0; //Признак создания архива

int Device;

int nk=0;// кол-во компонентов кодификатора

//-----------------------------------------------------------

void wait\_press\_key(char \*msg);

void WritelnString(char S[80]);

int Space(char \*s, int k);

int NotSpace(char \*s, int k);

char \*FillString(char \*S, unsigned char len, unsigned char pk);

double GetNumber(double MinNumber, double MaxNumber,

char m1,char n1,char m2,char n2);

void UsesDevice();

int SearchKodif(int Kod, int nk);

void SortKodif(int nk);

BASEUNIT.C

#include "baseunit.h"

//-----------------------------------------------------------

//Вывод сообщения на экран и ожидание нажатия клавиши

void wait\_press\_key(char \*msg)

{

fflush(stdin);

printf(msg);

getch();

}

//-----------------------------------------------

//Вывод строки S на экран, и в файл

void WritelnString(char S[80])

{

printf("%s\n",S);

if (Device==1)

fprintf(fRes,"%s\n",S);

}

//-----------------------------------------------

//Поиск в строке s пробела, начиная с позиции k

//Функция возвращает -1, если пробел не найден

int Space(char \*s, int k)

{ int i=k, pos=-1;

while(i<strlen(s) && s[i]!=' ')i++;

if (i<strlen(s)) pos=i;

return pos;

}

//-----------------------------------------------

//Поиск в строке s символа, отличного от пробела, с позиции k

//Функция возвращает -1, если такой символ не найден

int NotSpace(char \*s, int k)

{ int i=k, pos=-1;

while(i<strlen(s) && s[i]==' ')i++;

if (i<strlen(s)) pos=i;

return pos;

}

//-----------------------------------------------

// Дополнение строки S до длины len пробелами слева

// (pk=0) или справа (pk=1)

char \*FillString(char \*S, unsigned char len, unsigned char pk)

{

unsigned char i, n=strlen(S);

char St[35]="";

if (S[n-1]==' ') S[n-1]='\0';

n=strlen(S);

if (n<len)

{

for (i=1; i<len-n; i++ )

if (pk==0)

strcat(St," ");

else

strcat(S," ");

}

if (pk==0) {strcat(St,S); strncpy(S,&St[0],strlen(St));}

return S;

}

//-----------------------------------------------

// Ввод числа с клавиатуры; MinNumber,MaxNumber - допустимый

// диапазон; m1,n1,m2,n2 - формат сообщения о диапазоне

double GetNumber(double MinNumber, double MaxNumber,

char m1,char n1,char m2,char n2)

{

int k;

double Number;

unsigned char Cond=0;

char Format[80]="от %",s[10]="";

sprintf(s,"%d",m1);strcat(Format,s);strcat(Format,".");

sprintf(s,"%d",n1);strcat(Format,s);strcat(Format,"f до %");

sprintf(s,"%d",m2);strcat(Format,s);strcat(Format,".");

sprintf(s,"%d",n2);strcat(Format,s);strcat(Format,"f.");

do

{

Cond=1;

scanf("%lf",&Number);

k=0;

if (k!=0)

{

printf("\nНеправильный формат числа");

printf("\nПовторите ввод\n");

Cond=0;

}

else

if ((Number<MinNumber) || (Number>MaxNumber))

{

printf("Число должно быть в диапазоне ");

printf(Format, MinNumber,MaxNumber);

printf("\nПовторите ввод\n");

Cond=0;

}

}

while (Cond==0);

return Number;

}

//-----------------------------------------------

//Запрос об устройстве для вывода результатов

void UsesDevice()

{

printf("Укажите устройство для вывода результатов: \n");

printf(" 0 - только экран\n");

printf(" 1 - экран и текстовый файл\n");

Device=(int)ceil(GetNumber(0,1,1,0,1,0));

}

//-----------------------------------------------

//Двоичный поиск в массиве кодификатора KodifProducts

//элемента с кодом Kod

int SearchKodif(int Kod, int nk)

{

int i1,i2,m;

int SearchKod=-1;

i1=0; i2=nk-1;

while (i1<=i2)

{

m=(i1+i2)/2;

if (Kod==Kodifs[m].Kod)

{

SearchKod=m; break;

}

else

if (Kod>Kodifs[m].Kod)

i1=m+1;

else

i2=m-1;

}

return SearchKod;

}

//-----------------------------------------------

//-----------------------------------------------

//Сортировка кодификатора "школьным" методом

//по возрастанию кода продукта

void SortKodif(int nk)

{

unsigned char i,j;

int Kod;

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

{

for (j=0; j<nk; j++)

if (Kodifs[i].Kod<Kodifs[j].Kod)

{

Kodif=Kodifs[i];

Kodifs[i]=Kodifs[j];

Kodifs[j]=Kodif;

}

}

}

CheckInpFile.h

#include "desunit.h"

//-----------------------------------------------------------

extern FILE \*fArTxt;

extern FILE \*fArBin;

extern FILE \*fAddTxt;

extern FILE \*fKodif;

extern FILE \*fRes;

extern KodifType Kodif, //компонент кодификатора

Kodifs[180]; //массив компонентов кодификатора

typedef char string80[180];

typedef string80 FileStringAr[200];

typedef FileStringAr \*FileStringArPtr;

typedef ProductType ProductAr[200];

typedef ProductAr \*ProductArPtr;

FILE \*FileError; // файл протокола проверок }

char \*FileErrorName="Error.dat";

//-----------------------------------------------------------

int const NfMax = 200; // макс.кол-во строк в текстовом файле

int

np, // кол-во строк в файле FileProductInput }

nd, // кол-во строк в файле FileAddProduct }

nk, // кол-во строк в файле FileKodifProduct }

FatalError; // наличие ошибки в исходных данных }

string80 Sr; // сообщение об ошибке }

FileStringArPtr Sf; // массив строк текстового файла }

//--------------------------------------------------

//baseunit

extern void wait\_press\_key(char \*msg);

extern int Space(char \*s, int k);

extern int NotSpace(char \*s, int k);

extern void SortKodif(unsigned char nk);

extern char \*FillString(char \*S, unsigned char len, unsigned char pk);

extern int SearchKodif(int Kod, unsigned char nk);

//--------------------------------------------------

void ReadAndCheckSpaces(FILE \*F, char \*FileName, int \*nf,

int Nmax);

void FormatFileProduct(FILE \*F, char \*FileName, int \*nf,

int Nmax, ProductAr \*Prod);

void FormatFileKodif();

void ReportError1(char \*FileName, int i, int k, int d1,int d2);

void ReportError2(char \*FileName, int i, int k, double d1, double d2);

void CheckProdDiapason(char \*FileName, int nf, ProductAr \*Prod);

void CheckKodifDiapason();

void ProdParameters(ProductAr \*Prod, int n, char \*FileName);

void KodifParameters();

void ReadFileError();

void CheckFiles();

CheckInpFile.c

#include "CheckInpFile.h"

//-------------------------------------------------------------------------}

//Проверка форматов файлов "input.txt" и "add.txt"

void FormatFileProduct(FILE \*F, char \*FileName, int \*nf,

int Nmax, ProductAr \*Prod)

{

ProductType Product;

char k,k1,k2,n,Cond;

char \*p;

int i,Code;

string80 Sa,Sb;

ReadAndCheckSpaces(F,FileName,nf,Nmax);

if (FatalError) return;

for (i=0; i<(\*nf); i++)

{

strcpy(Sa,(\*Sf)[i]);

k=strlen(Sa);

if (Sa[k-1]==10) Sa[k-1]='\0';

Cond=1; k2=-1; k=0;

while (Cond)

{

k1=NotSpace(Sa,k2+1);

if (k1==-1)

Cond=0;

else

{

k2=Space(Sa,k1+1);

if (k2==-1) { k2=strlen(Sa)+1; Cond=0; }

k++;

if (k>8)

{

FatalError=1;

sprintf(Sr,"Файл %s : в строке %d свыше 8 элементов\n",FileName,i+1);

fwrite(Sr,sizeof(string80),1,FileError);

}

n=k2-k1; //длина текущего слова

strncpy(Sb,&Sa[k1],n);//копируем текущее слово в Sb

Sb[n]='\0'; //и добавляем символ '\0'.

switch (k)

{

case 1: Product.Kod=strtol(Sb,&p,10); break;

case 2:

if (strlen(Sb)>15)

{ FatalError=1;

sprintf(Sr,

"Файл %s : в строке %d длина элемента 3 свыше 15 символов\n",

FileName,i+1);

fwrite(Sr,sizeof(string80),1,FileError);

}

strncpy(Product.Family,&Sb[0],n);

Product.Family[n]='\0';

FillString(Product.Family,15,1); break;

case 3: if (strlen(Sb)>15)

{ FatalError=1;

sprintf(Sr,

"Файл %s : в строке %d длина элемента 3 свыше 20 символов\n",

FileName,i+1);

fwrite(Sr,sizeof(string80),1,FileError);

}

strncpy(Product.Adress,&Sb[0],n);

Product.Adress[n]='\0';

FillString(Product.Adress,15,1); break;

case 4: if (strlen(Sb)>10)

{ FatalError=1;

sprintf(Sr,

"Файл %s : в строке %d длина элемента 4 свыше 20 символов\n",

FileName,i+1); fwrite(Sr,sizeof(string80),1,FileError);}strncpy(Product.Remont,&Sb[0],n);

Product.Remont[n]='\0';

FillString(Product.Remont,10,1); break;

case 5:Product.Data1=strtol(Sb,&p,10); break;

case 6:Product.Data2=strtol(Sb,&p,10); break;

case 7:Product.Data3=strtol(Sb,&p,10); break;

case 8:Product.Data4=strtol(Sb,&p,10); break;

}

if((\*p)&&((k==5) || (k==6) || (k==7)|| (k==8)|| (k==1)))

{

FatalError=1;

sprintf(Sr,

"Файл %s : в строке %d неправильный формат элемента %d (%s)\n",

FileName,i+1,k,Sb);

fwrite(Sr,sizeof(string80),1,FileError);

}

}

}

if (k<8)

{

FatalError=1;

sprintf(Sr,"Файл %s : в строке %d меньше 8 элементов\n",FileName,i+1);

fwrite(Sr,sizeof(string80),1,FileError);

}

(\*Prod)[i]=Product;

}

}

//-------------------------------------------------------------------------

//Проверка наличия и ввод текстового файла, контроль количества

//строк, и удаление в нем пустых строк

void ReadAndCheckSpaces(FILE \*F, char \*FileName, int \*nf,

int Nmax)

{

char i,j,k;

char SignSpace; //{ признак пустой строки в файле }

string80 sw;

//{ Проверка наличия файла F с именем FileName }

if((F=fopen(FileName,"rt"))==NULL)

{

sprintf(Sr,"Отсутствует исходный файл %s\n",FileName);

FatalError=1;

fwrite(Sr,sizeof(string80),1,FileError);

return;

}

//{ Чтение текстового файла }

\*nf=0;

while (fgets(sw,80,F)!=NULL)

{

if ((\*nf)<200)

strcpy((\*Sf)[\*nf],sw);

(\*nf)++;

}

fclose(F);

//{ Проверка: является ли файл F пустым }

if (\*nf==0)

{

FatalError=1;

sprintf(Sr,"Исходный файл %s пустой\n",FileName);

fwrite(Sr,sizeof(string80),1,FileError);

return;

}

//{ Проверка: превышает ли кол-во строк макс.значение }

if (\*nf>NfMax)

{

FatalError=1;

sprintf(Sr,"В исходном файле %s свыше %d строк (%d)\n",FileName,NfMax,\*nf);

fwrite(Sr,sizeof(string80),1,FileError);

return;

}

//{ Удаление пустых строк из состава файла F }

SignSpace=0;

i=0;

while (i<(\*nf))

{

k=NotSpace((\*Sf)[i],0);

if ((k==-1) || ((\*Sf)[i][k]==10))

{

for (j=i; j<(\*nf)-1; j++)

strcpy((\*Sf)[j],(\*Sf)[j+1]);

(\*nf)--;

SignSpace=1;

}

else

i++;

}

if (SignSpace)

{

if((F=fopen(FileName,"wt"))==NULL)

{

sprintf(Sr,"Не удалось создать файл %s для удаления пустых строк\n",FileName);

FatalError=1;

fwrite(Sr,sizeof(string80),1,FileError);

return;

}

for (i=0; i<(\*nf);i++)

fputs((\*Sf)[i],F);

fclose(F);

}

} // ReadAndCheckSpaces };

//-------------------------------------------------------------------------}

//Проверка форматов файла "Kodif.txt"

void FormatFileKodif()

{

char k,k1,k2,n,Cond;

int i, Code;

string80 Sa,Sb,S1;

ReadAndCheckSpaces(fKodif,fKodifName,&nk,NfMax);

if (FatalError) return;

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

{

strcpy(Sa,(\*Sf)[i]);

k=strlen(Sa);

if (Sa[k-1]==10) Sa[k-1]='\0';

Cond=1; k2=-1; k=0;

while (Cond)

{

k1=NotSpace(Sa,k2+1);

if (k1==-1) Cond=0;

else

{

k2=Space(Sa,k1+1);

if (k2==-1) {k2=strlen(Sa)+1; Cond=0;}

k++;

if (k>3)

{

FatalError=1;

sprintf(Sr,"Файл Kodif.txt : в строке %d свыше 2 элементов\n",i+1);

fwrite(Sr,sizeof(string80),1,FileError);

}

n=k2-k1; //длина текущего слова

strncpy(Sb,&Sa[k1],n);//копируем текущее слово в Sb

Sb[n]='\0'; //и добавляем символ '\0'.

switch (k)

{

case 1:Code=sscanf\_s(Sb,"%d",&Kodif.Kod);

if (Code<1)

{

FatalError=1;

sprintf(Sr,

"Файл Kodif.txt : в строке %d неправильный формат элемента 1 (%s)\n",

i+1,Sb);

fwrite(Sr,sizeof(string80),1,FileError);

} break;

case 2:

if (strlen(Sb)>35)

{

FatalError=1;

sprintf(Sr,

"Файл Kodif.txt : в строке %d длина элемента 2 свыше 35 символов\n",

i+1);

fwrite(Sr,sizeof(string80),1,FileError);

}

strncpy(Kodif.Name,&Sb[0],n); //strlen(Sb));

Kodif.Name[n]=' '; Kodif.Name[n+1]='\0'; break;

case 3: sprintf(S1,"%s%s%s",Kodif.Name," ",Sb);

if (strlen(S1)>35)

{

FatalError=1;

sprintf(Sr,

"Файл Kodif.txt : в строке %d длина элемента 2 свыше 35 символов\n",

i+1);

fwrite(Sr,sizeof(string80),1,FileError);

}

strcpy(Kodif.Name,S1); FillString(Kodif.Name,35,1); break;

}

}

}

if (k<2)

{

FatalError=1;

sprintf(Sr,"Файл Kodif.txt : в строке %d меньше 2 элементов\n",i+1);

fwrite(Sr,sizeof(string80),1,FileError);

}

Kodifs[i]=Kodif;

}

} // FormatFileKodif };

//{-------------------------------------------------------------------------}

//Проверка параметров записей в файлах "input.txt" и "add.txt"

void ProdParameters(ProductAr \*Prod, int n, char \*FileName)

{

const char Measurs[2][5] = {{"шт. "},{"кг "}};

int i,j,k,Kod,Cond;

char Meas[5];

//{ Проверка дублирования параметра KodProduct }

for (i=0; i<n-1; i++)

{

Kod=(\*Prod)[i].Kod;

for (j=i+1; j<n; j++)

if (Kod==(\*Prod)[j].Kod)

{

FatalError=1;

sprintf(Sr,

"Файл %s : равные значения KodProduct в строках %d и %d (%d)\n",

FileName,i+1,j+1,Kod);

fwrite(Sr,sizeof(string80),1,FileError);

}

}

// { Проверка наличия параметра KodProduct в кодификаторе }

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

{

Kod=(\*Prod)[i].Kod;

k=SearchKodif(Kod,nk);

if (k==-1)

{

FatalError=1;

sprintf(Sr,

"Файл %s : код изделия %d (строка %d) отсутствует в кодификаторе\n",

FileName,Kod,i+1);

fwrite(Sr,sizeof(string80),1,FileError);

}

}

}

//{-------------------------------------------------------------------------}

//Проверка дублирования параметра KodProduct в файле "Kodif.txt"

void KodifParameters()

{

int i,j,Kod;

for (i=0; i<nk-1; i++)

{

Kod=Kodifs[i].Kod;

for (j=i+1; j<nk; j++)

if (Kod==Kodifs[j].Kod)

{

FatalError=1;

sprintf(Sr,

"Файл Kodif.txt: равные значения KodProduct в строках %d и %d (%d)\n",

i+1,j+1,Kod);

fwrite(Sr,sizeof(string80),1,FileError);

}

}

}

//{-------------------------------------------------------------------------}

//Чтение и вывод на экран протокола контроля текстовых файлов

void ReadFileError()

{

char i=1;

rewind(FileError);

while (i!=EOF)

{

i=fread(Sr,sizeof(string80),1,FileError);

if (i<1) i = EOF;

else printf("%s",Sr);

}

wait\_press\_key("\nДля продолжения нажмите любую клавишу\n");

} //ReadFileError

void CheckFiles()

{

ProductArPtr Products,

AddProducts;

SetConsoleOutputCP(1251);//Использует для вывода кодовую таблицу 1251

SetConsoleCP(1251);

if((FileError=fopen(FileErrorName,"wb+"))==NULL)

{

printf("\nФаил %s не найден\n",FileErrorName);

wait\_press\_key("\nДля завершения программы нажмите любую клавишу\n");

exit(0);

}

sprintf(Sr,"\n ПРОТОКОЛ КОНТРОЛЯ ТЕКСТОВЫХ ФАЙЛОВ\n");

fwrite(Sr,sizeof(string80),1,FileError);

FatalError=0;

Sf=(FileStringAr \*)malloc(sizeof(FileStringAr));

Products=(ProductAr \*)malloc(sizeof(ProductAr));

AddProducts=(ProductAr \*)malloc(sizeof(ProductAr));

FormatFileProduct(fArTxt,fArTxtName,&np,NfMax,

Products);

FormatFileProduct(fAddTxt,fAddTxtName,&nd,NfMax,

AddProducts);

FormatFileKodif();

if (! FatalError)

{

KodifParameters();

SortKodif(nk);

ProdParameters(Products,np,fArTxtName);

ProdParameters(AddProducts,nd,fAddTxtName);

}

if (FatalError)

sprintf(Sr,"\nСкорректируйте исходные файлы\n");

else

sprintf(Sr,"\nВ исходных файлах ошибок не обнаружено\n");

fwrite(Sr,sizeof(string80),1,FileError);

ReadFileError();

free(Sf); Sf=NULL;

free(Products); Products=NULL;

free(AddProducts); AddProducts=NULL;

fclose(FileError);

}

CreateArchKodif.h

#include "desunit.h"

//-----------------------------------------------------------

extern FILE \*fArTxt;

extern FILE \*fArBin;

extern FILE \*fAddTxt;

extern FILE \*fKodif;

extern FILE \*fRes;

extern unsigned char SignArchive;

extern KodifType Kodif, //компонент кодификатора

Kodifs[50]; //массив компонентов кодификатора

extern int nk; // кол-во компонентов кодификатора

//-----------------------------------------------------------

//baseunit

extern void wait\_press\_key(char \*msg);

extern int Space(char \*s, int k);

extern int NotSpace(char \*s, int k);

extern char \*FillString(char \*S, unsigned char len, unsigned char pk);

extern void WritelnString(char S[80]);

extern void SortKodif(int nk);

//-----------------------------------------------------------

int ReadProduct(FILE \*f, ProductType \*Product);

void CreateArchive();

int PrintArchive();

void MakeKodifs(int \*nk);

int PrintKodif(int nk);

CreateArchKodif.c

#include "CreateArchKodif.h"

//---------------------------------------------

//Считывается строка из указанного файла, производится ее разбор

//и формирование структуры Product. Возвращает 0

//Если найден конец файла - возвращает 1

int ReadProduct(FILE \*f,ProductType \*Product)

{

char sIn[80];

char slovo[80];

unsigned char Cond=0;

int k1,k2,k,i,n;

if ( fgets(sIn, 80, f) != NULL )

{

k=strlen(sIn);

if (sIn[k-1]==10) sIn[k-1]='\0';

i=0; k=0; Cond=1; k2=-1;

while(Cond)

{

k1=NotSpace(sIn,k2+1); //Определяем позицию начала слова

//Если k1>-1, найдено начало слова, иначе строка пустая или заканчивалась пробелами

if(k1==-1) Cond = 0;

else

{

k2=Space(sIn,k1+1);

if (k2==-1) {k2=strlen(sIn); Cond = 0;}

n=k2-k1; //длина текущего слова

strncpy(slovo,&sIn[k1],n); //копируем текущее слово в slovo

slovo[n]='\0'; //и добавляем символ '\0'.

k++; //k-номер найденного слова

switch ( k )

{

case 1:sscanf\_s(slovo,"%d",&Product->Kod); break;

case 2:strncpy(Product->Family,&slovo[0],n);

Product->Family[n]='\0';

FillString(Product->Family,15,1) ; break;

case 3:strncpy(Product->Adress,&slovo[0],n);

Product->Adress[n]='\0';

FillString(Product->Adress,15,1); break;

case 4:strncpy(Product->Remont,&slovo[0],n);

Product->Remont[1]='\0';

FillString(Product->Remont,1,1); break;

case 5:sscanf\_s(slovo,"%d",&Product->Data1); break;

case 6:sscanf\_s(slovo,"%d",&Product->Data2); break;

case 7:sscanf\_s(slovo,"%d",&Product->Data3); break;

case 8:sscanf\_s(slovo,"%d",&Product->Data4); break;

}

}

}

return 0;

}

else return 1;

}

//-----------------------------------------------

//Считываются строки описания продукта из текстового файла

//Формируется бинарный файл архива. Из текстового файла считывается

//кодификатор и записывается в массив. Массив сортируется и сохраняется

//до закрытия приложения. nk - количество элементов в массиве

void CreateArchive()

{

ProductType Product;

if((fArTxt=fopen(fArTxtName,"rt"))==NULL)

{

printf("\nФаил %s не найден\n",fArTxtName);

wait\_press\_key("\nДля завершения программы нажмите любую клавишу\n");

exit(0);

}

if((fArBin=fopen(fArBinName,"wb"))==NULL)

{

printf("\nФаил %s не создан\n",fArTxtName);

wait\_press\_key("\nДля завершения программы нажмите любую клавишу\n");

exit(0);

}

do

{

if (ReadProduct(fArTxt,&Product)==0)

fwrite(&Product,sizeof(ProductType),1,fArBin);

}

while ( ! feof(fArTxt) );

fclose(fArTxt);

fclose(fArBin);

MakeKodifs(&nk);

SortKodif(nk);

SignArchive=1;

printf("\n Архив создан\n");

wait\_press\_key("\nДля продолжения нажмите любую клавишу\n");

}

//-----------------------------------------------

//Из бинарного файла считывается архив и печатается на экране, в файл

int PrintArchive()

{

ProductType Product;

int i,p=1;

char s[100];

char slovo[80]="";

if (SignArchive==0)

{

printf("\nАрхив не создан. Режим отменяется.\n");

wait\_press\_key("\nДля продолжения нажмите любую клавишу\n");

return 0;

}

if((fArBin=fopen(fArBinName,"rb"))==NULL)

{

printf("\nФаил %s не найден. Режим отменяется.\n",fArBinName);

wait\_press\_key("\nДля продолжения нажмите любую клавишу\n");

SignArchive=0;

return 0;

}

WritelnString("\n АРХИВ СВЕДЕНИЙ О ВЫПУСКАЕМОЙ ПРОДУКЦИИ");

WritelnString(" -------------------------------------------------------------------------------------");

WritelnString("|К | Ф.И.О | Улица |Раз| План | Факт |");

WritelnString("|О | | |мер| ---------------------- |------------------------|");

WritelnString("|Д | | | | 1 | 2 | 1 | 2 |");

WritelnString("|------------------------------------------------------------------------------------|");

i=0;

if( p=fread(&Product,sizeof(ProductType),1,fArBin)<1) p=EOF;

while (p!=EOF)

{

i++; s[0]='|';s[1]='\0';

// sprintf(slovo,"%3d",i);strcat(s,slovo);strcat(s,". ");

sprintf(slovo,"%2d",Product.Kod);strcat(s,slovo);strcat(s," ");

strcat(s,Product.Family);

strcat(s,Product.Adress);

strcat(s,Product.Remont); strcat(s," ");

sprintf(slovo,"%02d.%02d.%4d",Product.Data1%100,(Product.Data1/100)%100,Product.Data1/10000);strcat(s,slovo);strcat(s," ");

sprintf(slovo,"%02d.%02d.%4d",Product.Data2%100,(Product.Data2/100)%100,Product.Data2/10000);strcat(s,slovo);strcat(s," ");

sprintf(slovo,"%02d.%02d.%4d",Product.Data3%100,(Product.Data3/100)%100,Product.Data3/10000);strcat(s,slovo);strcat(s," ");

sprintf(slovo,"%02d.%02d.%4d",Product.Data4%100,(Product.Data4/100)%100,Product.Data4/10000);strcat(s,slovo);strcat(s,"|");

WritelnString(s);

if( p=fread(&Product,sizeof(ProductType),1,fArBin)<1) p=EOF;

}

WritelnString("|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\n");

fclose(fArBin);

wait\_press\_key("\nДля продолжения нажмите любую клавишу\n");

return 0;

}

//-----------------------------------------------

// Чтение файла "Kodif.txt" и формирование массива Kodifs

void MakeKodifs(int \*nk)

{

char Sa[80],Sb[80];

unsigned char n,k,Cond;

int k1,k2;

if((fKodif=fopen(fKodifName,"rt"))==NULL)

{

printf("\nФаил %s не найден\n",fArTxtName);

wait\_press\_key("\nДля завершения программы нажмите любую клавишу\n");

exit(0);

}

\*nk=0;

while (fgets(Sa,80,fKodif)!=NULL)

{

n=strlen(Sa);

if (Sa[n-1]==10) Sa[n-1]='\0';

Cond=1; k2=-1; k=0;

while (Cond)

{

k1=NotSpace(Sa,k2+1);

if (k1==-1) Cond=0;

else

{

k2=Space(Sa,k1+1);

if (k2==-1) { k2=strlen(Sa); Cond=0; }

k++;

n=k2-k1;

strncpy(Sb,&Sa[k1],n);//копируем текущее слово в Sb

Sb[n]='\0'; //и добавляем символ '\0'.

switch ( k)

{

case 1:sscanf\_s(Sb,"%d",&Kodif.Kod); break;

case 2:strncpy(Kodif.Name,&Sb[0],n);

Kodif.Name[n]=' '; Kodif.Name[n+1]='\0'; break;

case 3:strcat(Kodif.Name,Sb); break;

}

}

}

k=strlen(Kodif.Name);

if ( Kodif.Name[k-1]==10 ) Kodif.Name[k-1]='\0';

FillString(Kodif.Name,15,1);

Kodifs[\*nk]=Kodif;

(\*nk)++;

}

fclose(fKodif);

}

//-----------------------------------------------

// Вывод кодификатора изделий на экран, в файл

int PrintKodif(int nk)

{

char s[80];

char slovo[80]="";

unsigned char i,j=0;

if (SignArchive==0)

{

printf("\nАрхив не создан. Режим отменяется.\n");

wait\_press\_key("\nДля продолжения нажмите любую клавишу\n");

return 0;

}

WritelnString("\n КОДИФИКАТОР ВЫПУСКАЕМОЙ ПРОДУКЦИИ");

WritelnString(" ----------------------------------------- ");

WritelnString("| Nп/п | Код | Номер телефона |");

WritelnString(" -----------------------------------------");

j=0;

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

{

Kodif=Kodifs[i];

j++; s[0]='|';s[1]='\0';

sprintf(slovo,"%4d",j);strcat(s,slovo);strcat(s,". ");

sprintf(slovo,"%6d",Kodif.Kod);strcat(s,slovo);strcat(s," ");

strcat(s,Kodif.Name);

strcat(s," |");

WritelnString(s);

}

WritelnString(" ----------------------------------------- ");

wait\_press\_key("\nДля продолжения нажмите любую клавишу\n");

return 0;

}

KURS.H

#include "desunit.h"

//-----------------------------------------------------------

extern FILE \*fRes;

extern int nk; // кол-во компонентов кодификатора

extern unsigned char SignArchive;

extern int Device;

//-----------------------------------------------------------

//baseunit

extern void wait\_press\_key(char \*msg);

extern void UsesDevice();

extern double GetNumber(double MinNumber, double MaxNumber,

char m1,char n1,char m2,char n2);

//CreateArchKodif

extern void CreateArchive();

extern int SortArchive();

extern int PrintArchive();

extern int PrintKodif(int nk);

extern void CheckFiles();

extern int AddArchive();

extern int DeleteArchive();

extern int ChangeArchive();

extern void WorkUpArchive();

KURS.C

#include "kurs.h"

//-----------------------------------------------------------

int main()

{

int KeyRegime; // ключ выбора режима работы

SetConsoleOutputCP(1251);

SetConsoleCP(1251);

UsesDevice();

if (Device==1)

if((fRes=fopen(fResName,"wt"))==NULL)

{

printf("\nФаил %s не создан\n",fResName);

wait\_press\_key("\nДля завершения программы нажмите любую клавишу\n");

exit(0);

}

SignArchive=0;

do

{

printf("\n Укажите режим работы программы:");

printf("\n 0 - конец работы;");

printf("\n 1 - контроль текстовых файлов;");

printf("\n 2 - создание архива изделий;");

printf("\n 3 - сортировка компонент архива;");

printf("\n 4 - печать архива изделий;");

printf("\n 5 - печать кодификатора изделий;");

printf("\n 6 - добавление компонент в архив;");

printf("\n 7 - удаление компоненты из архива;");

printf("\n 8 - изменение компоненты в архиве;");

printf("\n 9 - обработка архива \n");

fflush(stdin);

KeyRegime=(int)ceil(GetNumber(0,9,1,0,1,0));

switch (KeyRegime)

{

case 0 : ; break;

case 1 : CheckFiles(); break;

case 2 : CreateArchive(); break;

case 3 : SortArchive(); break;

case 4 : PrintArchive(); break;

case 5 : PrintKodif(nk); break;

case 6 : AddArchive(); break;

case 7 : DeleteArchive(); break;

case 8 : ChangeArchive(); break;

case 9 : WorkUpArchive(); break;

}

}

while (KeyRegime!=0);

if (Device==1)

fclose(fRes);

wait\_press\_key("\n Выход");

return 0;

}

AddArchiv.H

#include "desunit.h"

//-----------------------------------------------------------

extern FILE \*fArTxt;

extern FILE \*fAddTxt;

extern unsigned char SignArchive;

extern DynProduct \*Beg;

//-----------------------------------------------------------

//Baseunit

extern void wait\_press\_key(char \*msg);

//SortArchive

extern int ReadFileOut(int \*np, DynProduct \*\*L, DynProduct \*\*R);

extern int WriteFileOut(DynProduct \*L);

extern void DisposeProduct(DynProduct \*L);

//CreateArchKodif

extern int ReadProduct(FILE \*f, ProductType \*Product);

//--------------------------------------------------------

int AddArchive();

AddArchiv.C

#include "AddArchiv.h"

//-----------------------------------------------

//Добавление компонент в архив изделий

int AddArchive()

{

ProductType Product;

DynProduct \*Run; // текущий указатель дека архива

DynProduct \*L,\*R; // левый и правый указатели дека

unsigned char Cond;

int Sr, np;

if (SignArchive==0)

{

printf("\nАрхив не создан. Режим отменяется.\n");

wait\_press\_key("\nДля продолжения нажмите любую клавишу\n");

return 1;

}

if((fAddTxt=fopen(fAddTxtName,"rt"))==NULL)

{

printf("\nФаил %s не найден. Режим отменяется.\n",fAddTxtName);

wait\_press\_key("\nДля продолжения нажмите любую клавишу\n");

SignArchive=0;

return 1;

}

if ( ReadFileOut(&np,&L,&R) == 1) {fclose(fAddTxt); return 1;}

do

{

if (ReadProduct(fArTxt, &Product)==0)

{

Sr=Product.Kod;

Run=L; Cond=0;

while (Run!=NULL)

{

if (Run->Inf.Kod==Product.Kod)

{ Cond=1; break;}

Run=Run->Next;

}

if (Cond == 0 )

{

Run=(DynProduct \*)malloc(sizeof(DynProduct));

Run->Inf=Product;

R->Next=Run;

Run->Next=NULL;

R=Run;

}

else

printf("\nВ архиве уже есть изделие с кодом %6d",Sr);

}

}

while ( ! feof(fAddTxt) );

fclose(fAddTxt);

WriteFileOut(L);

printf("\nДополнение архива закончено\n");

wait\_press\_key("\nДля продолжения нажмите любую клавишу\n");

return 0;

}

DeleteArchiv.h

#include "desunit.h"

//-----------------------------------------------------------

extern unsigned char SignArchive;

//--------------------------------------------------------

//baseunit

extern void wait\_press\_key(char \*msg);

extern double GetNumber(double MinNumber, double MaxNumber,

char m1,char n1,char m2,char n2);

//SortArchive

extern int ReadFileOut(int \*np, DynProduct \*\*L, DynProduct \*\*R);

extern int WriteFileOut(DynProduct \*L);

extern void DisposeProduct(DynProduct \*L);

//--------------------------------------------------------

int DeleteArchive();

DeleteArchiv.c

#include "DeleteArchiv.h"

//-----------------------------------------------

//Удаление компонента из архива изделий. Просмотр дека слева направо

int DeleteArchive()

{

int Kod,KodPr;

unsigned char Cond;

DynProduct \*L, \*R; // левый и правый указатели дека

DynProduct \*Del; // указатель на удаляемый элемент из дека архива

DynProduct \*Run; // текущий указатель дека архива

int np;

char Sr[80]="";

if ( ! SignArchive )

{

printf("\nАрхив не создан. Режим отменяется.\n");

wait\_press\_key("\nДля продолжения нажмите любую клавишу\n");

return 1;

}

ReadFileOut(&np,&L,&R);

printf("\nУкажите код удаляемого компонента : ");

Kod=(int)ceil(GetNumber(0,999999,1,0,6,0));

Cond=0; KodPr=Kod;

if ( Kod == L->Inf.Kod)

{

Cond=1;

Del=L; L=L->Next;

free(Del);

}

else

{

Run=L;

while ( Run->Next != NULL )

{

if ( Kod == Run->Next->Inf.Kod )

{

Cond=1;

Del=Run->Next;

if (Run->Next->Next == NULL )

{Run->Next=NULL;}

else

{Run->Next=Run->Next->Next;}

free(Del);

break;

}

Run=Run->Next;

}

}

if( Cond == 1 )

{

np--;

WriteFileOut(L);

printf("\nУдаление компонента из архива закончено");

}

else

printf("\nВ архиве нет компонента с кодом %6d",KodPr);

wait\_press\_key("\nДля продолжения нажмите любую клавишу\n");

return 0;

}

SortArchiv.h

#include "desunit.h"

//-----------------------------------------------------------

extern FILE \*fArBin;

extern unsigned char SignArchive;

//-----------------------------------------------------------

//--------------------------------------------------------

//baseunit

extern void wait\_press\_key(char \*msg);

//--------------------------------------------------------

int SortArchive();

int ReadFileOut(int \*np, DynProduct \*\*L, DynProduct \*\*R);

int WriteFileOut(DynProduct \*L);

void DisposeProduct(DynProduct \*L);

SortArchiv.c

#include "SortArchiv.h"

//-----------------------------------------------

//Сортировка архива по возрастанию кода изделия. Просмотр дека

//с левой стороны и сортировка по убыванию "школьным" методом.

//При просмотре дека с правой стороны сортировка будет по возрастанию

int SortArchive()

{

int np,i,j;

ProductType Product;

DynProduct \*Runi;

DynProduct \*Runj; // текущие указатели дека архива

DynProduct \*L,\*R; // левый и правый указатели дека

if ( ! SignArchive )

{

printf("\nАрхив не создан. Режим отменяется.\n");

wait\_press\_key("\nДля продолжения нажмите любую клавишу\n");

return 1;

}

ReadFileOut(&np,&L,&R);

for ( Runi=L; Runi!=NULL; Runi=Runi->Next )

for ( Runj=L; Runj!=NULL; Runj=Runj->Next )

if ( Runi->Inf.Kod < Runj->Inf.Kod )

{

Product=Runi->Inf; Runi->Inf=Runj->Inf;

Runj->Inf=Product;

}

WriteFileOut(L);

printf("\nСортировка архива закончена.\n");

wait\_press\_key("\nДля продолжения нажмите любую клавишу\n");

return 0;

} //-----------------------------------------------

//Чтение из бинарного файла архива и формирование дека записей

//Включение элементов в дек производится с левой стороны

int ReadFileOut(int \*np, DynProduct \*\*L, DynProduct \*\*R)//, DynProduct \*\*Rp)

{

ProductType Product;

DynProduct \*Run; // текущий указатель дека архива

int p;

if (SignArchive==0)

{

printf("\nАрхив не создан. Режим отменяется.\n");

wait\_press\_key("\nДля продолжения нажмите любую клавишу\n");

return 1;

}

if((fArBin=fopen(fArBinName,"rb"))==NULL)

{

printf("\nФаил %s не найден. Режим отменяется.\n",fArBinName);

wait\_press\_key("\nДля продолжения нажмите любую клавишу\n");

SignArchive=0;

return 1;

}

\*np=0;

\*L = NULL; \*R=NULL;

if( p=fread(&Product,sizeof(ProductType),1,fArBin)<1) p=EOF;

while (p!=EOF)

{

Run=(DynProduct \*)malloc(sizeof(DynProduct));

(\*np)++;

Run->Inf=Product;

Run->Next=\*L;

\*L=Run;

if (\*R==NULL)

\*R=\*L;

if( p=fread(&Product,sizeof(ProductType),1,fArBin)<1) p=EOF;

}

fclose(fArBin);

return 0;

return 0;

}

//-----------------------------------------------

//Пересылка записей из дека в бинарный файл архива

//Просмотр с правой стороны с последующим удалением дека

int WriteFileOut(DynProduct \*L)

{

ProductType Product;

DynProduct \*Run; // текущий указатель дека архива

if((fArBin=fopen(fArBinName,"wb"))==NULL)

{

printf("\nФаил %s не создан\n",fArBinName);

wait\_press\_key("\nДля завершения программы нажмите любую клавишу\n");

exit(0);

}

Run=L;

while ( Run != NULL)

{

Product=Run->Inf;

fwrite(&Product,sizeof(ProductType),1,fArBin);

Run=Run->Next;

}

fclose(fArBin);

DisposeProduct(L);

return 0;

}

//-----------------------------------------------

//Удаление дека изделий с правой стороны

void DisposeProduct(DynProduct \*L)

{

DynProduct \*Run; // текущий указатель дека архива

while (L != NULL)

{

Run=L;

L=L->Next;

free(Run);

}

}

EditArchiv.h

#include "desunit.h"

//-----------------------------------------------------------

extern unsigned char SignArchive;

extern DynProduct \*L,\*R; // левый и правый указатели дека

//-----------------------------------------------------------

//------------------------------------------------------

//baseunit

extern void wait\_press\_key(char \*msg);

extern int Space(char \*s, int k);

extern int NotSpace(char \*s, int k);

extern char \*FillString(char \*S, unsigned char len, unsigned char pk);

extern double GetNumber(double MinNumber, double MaxNumber,

char m1,char n1,char m2,char n2);

//SortArchive

extern int ReadFileOut(int \*np, DynProduct \*\*L, DynProduct \*\*R);

extern int WriteFileOut(DynProduct \*L);

extern void DisposeProduct(DynProduct \*L);

//------------------------------------------------------

int ChangeArchive();

void MakeComponent(ProductType \*Product);

EditArchiv.c

#include "EditArchiv.h"

//-----------------------------------------------

// Изменение компонента в архиве изделий

// Просмотр дека справа налево

int ChangeArchive()

{

int Kod,Kod1,np;

unsigned char Cond=0;

ProductType Product;

DynProduct \*L,\*R; // левый и правый указатели очереди

DynProduct \*Run; // текущий указатель очереди архива

if ( ! SignArchive )

{

printf("\nАрхив не создан. Режим отменяется.\n");

wait\_press\_key("\nДля продолжения нажмите любую клавишу\n");

return 1;

}

ReadFileOut(&np,&L,&R);

printf("\nУкажите код изделия изменяемого компонента :");

Kod=(int)ceil(GetNumber(0,999999,1,0,6,0));

Kod1=Kod; printf("Kod = %2d",Kod1);

Run=L; Cond=0;

while (Run!=NULL)

{

if (Kod==Run->Inf.Kod)

{

Cond=1;

Product=Run->Inf;

printf("\n Укажите следующие реквизиты :\n");

printf(" план-1 план-2 факт-1 факт-2\n");

MakeComponent(&Product);

Run->Inf=Product;

WriteFileOut(L);

printf("Изменение компонента в архиве закончено\n");

break;

}

Run=Run->Next;

}

if (! Cond )

printf("\nВ архиве нет компонента с кодом %d\n",Kod1);

wait\_press\_key("\nДля продолжения нажмите любую клавишу\n");

return 0;

}

//-----------------------------------------------

// Формирование изменяемого компонента

void MakeComponent(ProductType \*Product)

{

unsigned char k,n,Cond;

int k1,k2;

char Sa[80],Sb[80];

printf("Тек.зн. %8d %8d %8d %8d\n",

Product->Data1,Product->Data2,Product->Data3,Product->Data4);

fflush(stdin);

gets(Sa);

Cond=1; k2=-1; k=0;

while (Cond==1)

{

k1=NotSpace(Sa,k2+1);

if (k1==-1) Cond=0;

else

{

k2=Space(Sa,k1+1);

if (k2==-1) { k2=strlen(Sa)+1; Cond=0; }

k++; n=k2-k1;

strncpy(Sb,&Sa[k1],n);//копируем текущее слово в Sb

Sb[n]='\0'; //и добавляем символ '\0'.

switch ( k )

{

case 1:sscanf\_s(Sb,"%d",&Product->Data1); break;

case 2:sscanf\_s(Sb,"%d",&Product->Data2); break;

case 3:sscanf\_s(Sb,"%d",&Product->Data3); break;

case 4:sscanf\_s(Sb,"%d",&Product->Data4); break;

}}}}

ReportArchiv.h

#include "desunit.h"

//-----------------------------------------------------------

extern unsigned char SignArchive;

extern unsigned char nk; // кол-во компонентов кодификатора

extern KodifType Kodifs[50]; //массив компонентов кодификатора

//-----------------------------------------------------------

//------------------------------------------------------

//baseunit

extern void wait\_press\_key(char \*msg);

extern double GetNumber(double MinNumber, double MaxNumber,

char m1,char n1,char m2,char n2);

extern void WritelnString(char S[80]);

extern int SearchKodif(int Kod, unsigned char nk);

//SortArchive

extern int ReadFileOut(int \*np, DynProduct \*\*L, DynProduct \*\*R);

extern void DisposeProduct(DynProduct \*L);

//------------------------------------------------------

void WorkUpArchive();

int dataday(int data);

void print(ProductType Product);

void sort(int \*mindata,int \*minKod);

ReportArchiv.c

#include "ReportArchiv.h"

// Обработка архива изделий и формирование отчетов

void WorkUpArchive()

{

unsigned char Kodik; //{ номер цеха }

int j,k=-1,m,np,dataP,dataF;

int minKod[3],mindata[3]={999999,999999,999999};

DynProduct \*L,\*R; // левый и правый указатели дека

DynProduct \*Run;

ProductType Product;

WritelnString("Ведомость о заявителях,у которых ремонт выполнен с опозданием в три и более месяцев");

WritelnString(" -------------------------------------------------------------------------------------");

WritelnString("|К | Ф.И.О | Улица |Раз| План | Факт |");

WritelnString("|О | | |мер| ---------------------- |------------------------|");

WritelnString("|Д | | | | 1 | 2 | 1 | 2 |");

WritelnString("|------------------------------------------------------------------------------------|");

if ( ! SignArchive )

{

printf("Архивный файл не создан. Режим отменяется");

return;

}

ReadFileOut(&np,&L,&R);

Run=L;

while(Run!=NULL)

{ dataP=Run->Inf.Data2;

dataF=Run->Inf.Data4;

dataP=dataday(dataP);

dataF=dataday(dataF);

if (dataF-dataP>180)

print(Run->Inf);

Run=Run->Next;

}

WritelnString(" ------------------------------------------------------------------------------------ ");

WritelnString("Три первоочередника на мелкий ремонт");

WritelnString(" -------------------------------------------------------------------------------------");

WritelnString("|К | Ф.И.О | Улица |Раз| План | Факт |");

WritelnString("|О | | |мер| ---------------------- |------------------------|");

WritelnString("|Д | | | | 1 | 2 | 1 | 2 |");

WritelnString("|------------------------------------------------------------------------------------|");

Run=L;

while(Run!=NULL)

{if(! strcmp(Run->Inf.Remont,"м"))

{ dataP=dataday(Run->Inf.Data1);

if (dataP<mindata[0])

{mindata[2]=mindata[1];

minKod[2]=minKod[1];

mindata[1]=mindata[0];

minKod[1]=minKod[0];

mindata[0]=dataP;

minKod[0]=Run->Inf.Kod;

}

else

if (dataP<mindata[1])

{ mindata[2]=mindata[1];

minKod[2]=minKod[1];

mindata[1]=dataP;

minKod[1]=Run->Inf.Kod;

}

else

if (dataP<mindata[2])

{ mindata[2]=dataP;

minKod[2]=Run->Inf.Kod;}

sort(mindata,minKod);

}

Run=Run->Next;

}

Run=L;

while(Run!=NULL)

{ if (Run->Inf.Kod==minKod[0] || Run->Inf.Kod==minKod[1] || Run->Inf.Kod==minKod[2] )

print(Run->Inf);

Run=Run->Next;

}

WritelnString(" ------------------------------------------------------------------------------------ ");

DisposeProduct(L);

printf("\nОбработка архива закончена");

wait\_press\_key("\nДля продолжения нажмите любую клавишу\n");

} //WorkUpArchive

int dataday(int data)

{data=data%100+(data/100)%100\*30+(data/10000)\*365;

return data;}

void sort(int \*mindata, int \*minKod)

{int i,buf,bufkod;

for (i=0; i<2; i++)

if (mindata[i]>mindata[i+1])

{ buf=mindata[i]; bufkod=minKod[i];

mindata[i]=mindata[i+1]; minKod[i]=minKod[i+1];

mindata[i+1]=buf; minKod[i+1]=bufkod;

}

}

void print(ProductType Product)

{char s[100];

char slovo[80]="";

s[0]='|';s[1]='\0';

sprintf(slovo,"%2d",Product.Kod);strcat(s,slovo);strcat(s," ");

strcat(s,Product.Family);

strcat(s,Product.Adress);

strcat(s,Product.Remont); strcat(s," ");

sprintf(slovo,"%02d.%02d.%4d",Product.Data1%100,(Product.Data1/100)%100,Product.Data1/10000);strcat(s,slovo);strcat(s," ");

sprintf(slovo,"%02d.%02d.%4d",Product.Data2%100,(Product.Data2/100)%100,Product.Data2/10000);strcat(s,slovo);strcat(s," ");

sprintf(slovo,"%02d.%02d.%4d",Product.Data3%100,(Product.Data3/100)%100,Product.Data3/10000);strcat(s,slovo);strcat(s," ");

sprintf(slovo,"%02d.%02d.%4d",Product.Data4%100,(Product.Data4/100)%100,Product.Data4/10000);strcat(s,slovo);strcat(s,"|");

WritelnString(s);

}