Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего

образования «Севастопольский государственный университет»

Кафедра информационных систем

КУРСОВОЙ ПРОЕКТ

ПО ДИСЦИПЛИНЕ «ОСНОВЫ ПРОГРАММИРОВАНИЯ И

АЛГОРИТМИЧЕСКИЕ ЯЗЫКИ»

НА ТЕМУ:

«ПРОГРАММА ПОДСЧЁТА РАБОЧИХ ПО СТАЖУ И ЦЕХУ»

Пояснительная записка

RU.920101001.02463-01 81 12

Листов 75

Студента 2 курса группы ИС/б-21-о

направление подготовки 09.03.02

*(подпись)\_\_\_\_\_\_\_* Куркчи А.Э.

« » \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2015г.

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

(должность, ученое звание, фамилия и инициалы)

Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Члены комиссии \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (фамилия и инициалы)

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

(подпись) (фамилия и инициалы)

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

(подпись) (фамилия и инициалы)

г. Севастополь – 2015 г.

**АННОТАЦИЯ**

В документе описывается программа, предназначенная для подсчёта рабочих в цеху по их стажу. Она написанная в соответствии с постановкой задачи на курсовое проектирование по дисциплине "Основы программирования и алгоритмические языки".

Входными данными является файл, в котором хранится таблица с информацией о рабочих. Выходными данными – файл с таблицей, содержащей информацию о рабочих в каждом цеху по их стажу.

Программа была тщательно тестирована с различными вариантами входных данных и работает корректно. В качестве языка программирования был выбран *С+*+ и среда *NetBeans*. Программа работает под управлением ОС *Linux*.

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 4](#_Toc438167602)

[1. НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ ПРОГРАММЫ 5](#_Toc438167603)

[2. ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ 6](#_Toc438167604)

[2.1. Постановка задачи 6](#_Toc438167605)

[2.2. Входные данные 7](#_Toc438167606)

[2.3. Выходные данные 7](#_Toc438167607)

[2.4. Внутреннее представление программы 8](#_Toc438167608)

[2.5. Описание алгоритма программы 9](#_Toc438167609)

[2.6. Выбор языка программирования 20](#_Toc438167610)

[2.7. Обоснование состава технических и программных средств 20](#_Toc438167611)

[3. ВЫПОЛНЕНИЕ ПРОГРАММЫ 22](#_Toc438167612)

[4. ТЕСТОВЫЙ ПРИМЕР РАБОТЫ ПРОГРАММЫ 27](#_Toc438167613)

[ЗАКЛЮЧЕНИЕ 33](#_Toc438167614)

[БИБЛИОГРАФИЧЕСКИЙ СПИСОК 34](#_Toc438167615)

[Приложение А 35](#_Toc438167616)

Изм.

Лист

№ докум.

Подпись

Дата

Лист

3

КУРСОВОЙ ПРОЕКТ

Выполнил

Куркчи А.Э.

Провер.

Забаштанский А.К.

Н. Контр.

Утверд.

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

Лит.

Листов

75

Кафедра ИС группа ИС/б-21-о

ВВЕДЕНИЕ

В курсовом проекте на основании технического задания, разрабатывается программа обработки информации, представленной в табличной форме.

Входные данные программы хранятся во внешнем текстовом файле в виде таблицы. В процессе выполнения программы данные считываются в двунаправленный список, реализуется возможность просмотра, добавления и редактирования данных, производится обработка данных (подсчёта количества рабочих в цеху по их стажу) и выдача результатов, представленных также в виде таблицы. Взаимодействие пользователя с программой организуется в интерактивном режиме. Актуальность данной программы заключается в возможности ускорить процесс статистического исследования стажа рабочих по цехам.

В процессе выполнения курсового проекта была усовершенствована техника программирования на языке высокого уровня,

Курсовое проектирование включает следующие этапы:

- изучение математического метода решения задачи;

- определение технических условий и ограничений;

- разработка структурной схемы алгоритма;

- разработка тестового примера;

- написание текста программы;

- отладка программы;

- проведение вычислительного эксперимента;

- оформление технической документации на программу.

Разработка велась на основании задания по выполнению работы, кафедры Информационных систем Севастопольского государственного университета.

1. **НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ ПРОГРАММЫ**

Настоящая программа пример работы базы данных малого предприятия. В ней учитывается информация о рабочих: табельный номер, Ф.И.О., год рождения, пол, профессия, стаж, разряд, номер цеха, номер участка и зарплата. В результате работы программы выводится таблица, в которой подсчитывается количество рабочих в цехах по стажу.

Программа работает в текстовом режиме под управлением операционной системы *Linux*. Для загрузки списка рабочих требуется специально организованный файл. Вывод выходной таблицы производится как на экран, так и в файл по запросу пользователя. Для работы программы подходит практически любой *PC* компьютер, оснащенный цветным дисплеем.

Данная программа может применяться на малых предприятиях для анализа и статистической обработки количества рабочих по стажу.

1. **ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ**
   1. **Постановка задачи**

Требуется написать программу обработки статистической информации, осуществляющую ввод исходной информации из файла, необходимую ее организацию, обработку в соответствии с вариантом задания №12 методических указаний, а также вывести на экран или в файл выходную таблицу заданного вида, который представлен в таблице 2.1.

В соответствии с вариантом №12, дан входной файл следующей структуры: таблицы с шапкой; в таблице присутствуют следующие поля: порядковый номер, номер магазина (4 символа), номер секции (4 символа), номер чека (8 символов), название товара (до 15 символов), артикль товара (4 символа), цена товара (8 символов), количество товара (4 символа) и дата покупки (три поля с числом, месяцем и годом).

Таблица 2.1 – Вид выходной таблицы

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Цех | Количество рабочих по стажу лет | | | | | | Всего |
| До 6 | 6-11 | 11-15 | 16-20 | 21-25 | Свыше 26 |
|  |  |  |  |  |  |  |  |

Интерфейсная часть программы оформлена в виде контекстно-зависимого меню, главная часть которого имеет следующие пункты:

* «Добавление»
* «Сортировка»
* «Поиск»
* «Изменение»
* «Удаление»
* «Обработка»
* «Очистка»
* «Сохранение»
* «Загрузка»
* «Выход»
  1. **Входные данные**

Входными данными служит текстовый файл, расположенный в рабочей директории программы. Он представляет собой таблицу, составленную из псевдографических символов и соответствующую таблице 2.2.

Таблица 2.2 – вид входной таблицы

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *ID* | Имя | Год | Пол | Профессия | Опыт | Разряд | Цех | Участок | Зарплата |
| 1 | Степан Свердлов | 1963 | М | Машинист | 8 | 3 | 1 | 1 | 4997.52 |
| 2 | Сергей Меркулов | 1966 | М | Плотник | 10 | 1 | 2 | 1 | 2287.05 |
| 3 | Константин Дробыш | 1978 | М | Машинист | 6 | 1 | 10 | 3 | 3769.17 |
| 4 | Ефросиния Иванова | 1957 | Ж | Слесарь | 24 | 2 | 4 | 1 | 7175.92 |

Причём имя рабочего занимает максимум 20 символов, а профессия 10, табельный номер, год рождения, опыт, разряд, номер цеха, номер участка целого типа и имеют длину не более 4 символов, зарплата число с плавающей точкой, пол представляет собой символ М или Ж, для мужчин и женщин соответственно. Количество записей в файле ограничено только доступной оперативной памятью, все записи объединены в таблицу, нарисованную символами псевдографики.

* 1. **Выходные данные**

Выходные данные представляют собой таблицу, составленную путём обработки всех входных данных, каждая строка которой является отдельным цехом, в случае, когда рабочих в этом цеху нет, он игнорируется. Сформированная таблица 2.3 может быть сохранена в файл по запросу пользователя.

Таблица 2.3 – Пример выходной таблицы

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Цех | Количество рабочих по стажу лет | | | | | | Всего |
| До 6 | 6-11 | 11-15 | 16-20 | 21-25 | Свыше 26 |
| 1 | 0 | 3 | 1 | 4 | 3 | 2 | 13 |
| 2 | 3 | 2 | 2 | 0 | 0 | 3 | 10 |
| 3 | 0 | 0 | 1 | 1 | 0 | 4 | 6 |
| 4 | 1 | 0 | 0 | 3 | 2 | 2 | 8 |

* 1. **Внутреннее представление программы**

Программа реализована на языке высокого уровня *C++* в его стандартном виде. В качестве библиотеки для работы с консолью в *Linux* используется библиотека *NCURSES*. Средой программирования выбран *NetBeans* за его кроссплатформенность и простоту в использовании.

Библиотека *NCURSES* содержит в себе множество функций по манипуляции консолью, а также упрощает абстракцию при создании интерфейса, благодаря встроенным структурам «окон» и «панелей».

Выбранные средства являются универсальными для всех *Linux* систем, а библиотека *NCURSES* может быть заменена на аналогичную реализацию *CURSES*, установленную в системе.

Так как исходные данные по заданию могут включать в себя неизвестное заранее количество элементов, в качестве хранилища входной информации статические переменны использовать невозможно. Наиболее подходящим под задание является двунаправленный список, так как он позволяет переставлять элементы, искать в любом из направлений, вставлять и удалять элементы из любого места за оптимальное время.

Для хранения записей входного файла используется структура data\_t:

struct data\_t **{**

int id**;**

wchar\_t name **[**DATA\_NAME\_LEN**+**1**];**

int b\_year**;**

wchar\_t sex**;**

wchar\_t prof **[**DATA\_PROF\_LEN**+**1**];**

int exp**;**

int rank**;**

int guild**;**

int sect**;**

float salary**;**

**};**

В полях *id*, *b\_year*, *exp*, *rank*, *guild*, *sect* хранятся целочисленный значения типа *int*, не превышающие 231-1 для 32­-х разрядных систем, чего вполне достаточно для этого типа записей.

В поле *name* хранится имя длиной до 20 символов (21й символ – символ конца строки). Так как строка должна хранить названия на русском языке она должна вмещать символы кодировки *unicode*, для этого выбран тип *wchar\_t*, который представляет собой расширенный тип *char*, состоящий из 2х байт.

По аналогичной причине для поля *prof* выбран тип *wchar\_t* и длина 10 символов.

Программа не позволяет вводить некорректные данные и соответственно их сохранять.

* 1. **Описание алгоритма программы**
     1. Описание алгоритма основной программы

Структурная схема алгоритма программы приведена на чертеже 3 формата А1. В соответствии с алгоритмом программа функционирует следующим образом:

Блок 1. Устанавливаются основные параметры кодировки ввода-вывода и параметры обработки управляющих сигналов процесса;

Блок 2. Проверяется установлены ли настройки кодировки и управляющих сигналов;

Блок 3. Проверяется необходима ли генерация списка из начальных данных;

Блок 4. Вызывается функция генерации списка из начальных данных и запись списка в текстовый и бинарный файлы, после чего память, отведённая под список, освобождается;

Блок 5. Устанавливаются настройки отображения в консоли и определяются основные переменные, необходимые для работы программы;

Блок 6. Вызывается функция рисования интерфейса программы в консоли и обработки нажатой клавиши;

Блок 7. Проверка был ли запрошен выход из программы;

Блок 8. Высвобождается память и восстанавливаются настройки режима консоли.

* + 1. Описание алгоритма подпрограммы добавления элемента в список

Структурная схема алгоритма подпрограммы добавления элемента в список приведена на чертеже 1 формата А1. В соответствии с алгоритмом подпрограмма функционирует следующим образом:

Блок 1. Выделяется память под элемент списка и заполняется требуемыми данными;

Блок 2. Проверяется пуст ли список;

Блок 3. Устанавливается указатель на следующий элемент в последнем равным текущему элементу;

Блок 4. Устанавливается указатель на начало списка равным текущему элементу;

Блок 5. Устанавливается указатель на конец списка равным текущему элементу.

* + 1. Описание алгоритма подпрограммы обработки клавиш

Структурная схема алгоритма подпрограммы обработки клавиш приведена на чертеже 2 формата А1. В соответствии с алгоритмом подпрограмма функционирует следующим образом:

Блок 1. Происходит ввод клавиши;

Блок 2. Проверяется есть ли сообщение;

Блок 3. Закрывается сообщение, перерисовывается меню и данные;

Блок 4. Происходит выбор по клавише;

Блок 5. Показывается сообщение с информацией о программе и авторе;

Блок 6. Показывается/скрывается меню программы;

Блок 7. Проверяется отображается ли меню;

Блок 8. Обработка клавиши делегируется в блок данных;

Блок 9. Обработка клавиши делегируется в блок меню.

* + 1. Описание алгоритма подпрограммы рисования формы ввода

Структурная схема алгоритма подпрограммы рисования формы ввода приведена на рисунке 2.1.

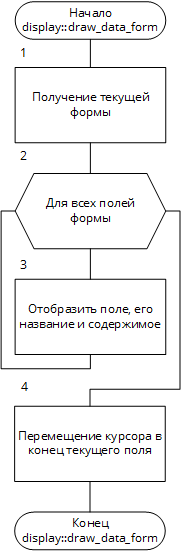


Рисунок 2.1 – Схема подпрограммы рисования формы ввода

В соответствии с алгоритмом подпрограмма функционирует следующим образом:

Блок 1. Получается информация о текущей форме;

Блок 2. Происходит итерация по всем полям формы;

Блок 3. Отображается поле, его название и введённое заранее содержимое;

Блок 4. Перемещается курсор в конец выделенного поля.

* + 1. Описание алгоритма подпрограммы освобождения списка

Структурная схема алгоритма подпрограммы освобождения списка приведена на рисунке 2.2.

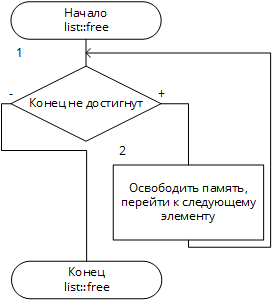


Рисунок 2.2 – Схема подпрограммы освобождения списка

В соответствии с алгоритмом подпрограмма функционирует следующим образом:

Блок 1. Проверяется достигнут ли конец списка;

Блок 2. Высвобождается память под текущим элементом и происходит переход к следующему.

* + 1. Описание алгоритма подпрограммы сохранения списка

Структурная схема алгоритма подпрограммы сохранения списка приведена на рисунке 2.3.

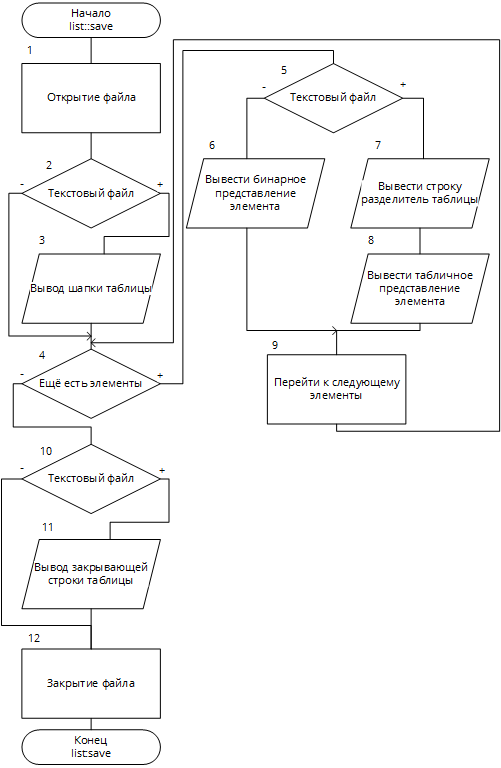


Рисунок 2.3 – Схема подпрограммы сохранения списка

В соответствии с алгоритмом подпрограмма функционирует следующим образом:

Блок 1. Открывается необходимый дескриптор файла;

Блок 2. Проверяется происходит ли сохранение в текстовый файл;

Блок 3. В файл выводится шапка таблицы;

Блок 4. Проверяется есть ли ещё элементы;

Блок 5. Проверяется происходит ли сохранение в текстовый файл;

Блок 6. В файл выводится бинарное представление текущего элемента;

Блок 7. В файл выводится разделитель строк таблицы;

Блок 8. В файл выводится табличное представление текущего элемента;

Блок 9. Происходит переход к следующему элементу списка;

Блок 10. Проверяется происходит ли сохранение в текстовый файл;

Блок 11. В файл выводится закрывающая строка таблицы;

Блок 12. Происходит закрытие дескриптора файла.

* + 1. Описание алгоритма подпрограммы создания сводной таблицы

Структурная схема алгоритма подпрограммы создания сводной таблицы приведена на рисунке 2.4.

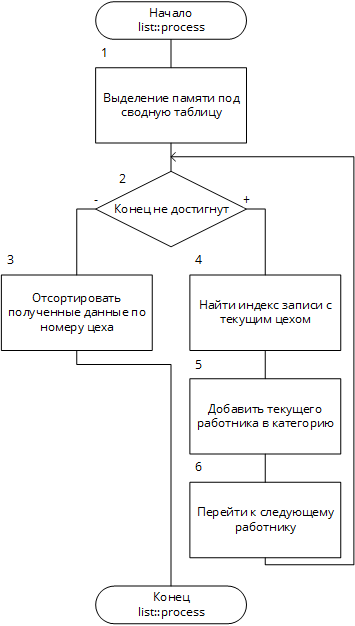


Рисунок 2.4 – Схема подпрограммы создания сводной таблицы

В соответствии с алгоритмом подпрограмма функционирует следующим образом:

Блок 1. Происходит выделение памяти под сводную таблицу;

Блок 2. Проверяется достигнут ли конец списка;

Блок 3. Полученные данные сортируются по номеру цеха в возрастающем порядке;

Блок 4. Находится индекс записи по текущему цеху;

Блок 5. Текущий работник добавляется в нужную категорию записи;

Блок 6. Происходит переход к следующему работнику.

* + 1. Описание алгоритма подпрограммы отображения сообщения

Структурная схема алгоритма подпрограммы отображения сообщения приведена на рисунке 2.5.



Рисунок 2.5 – Схема подпрограммы отображения сообщения

В соответствии с алгоритмом подпрограмма функционирует следующим образом:

Блок 1. Содержимое блока сообщения устанавливается на указанное и показывается панель с сообщением.

* + 1. Описание алгоритма подпрограммы обработки нажатия клавиши формы ввода

Структурная схема алгоритма подпрограммы обработки нажатия клавиши формы ввода приведена на рисунке 2.6.

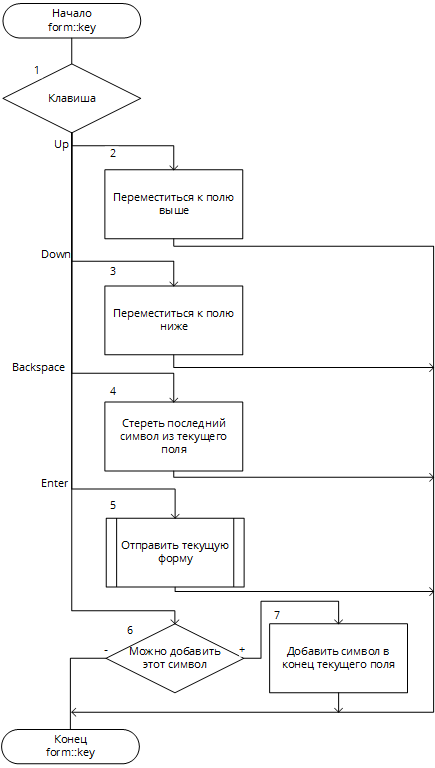


Рисунок 2.6 – Схема подпрограммы обработки нажатия клавиши формы ввода

В соответствии с алгоритмом подпрограмма функционирует следующим образом:

Блок 1. Происходит выбор нажатой клавиши;

Блок 2. Происходит перемещение указателя к полю, расположенному выше;

Блок 3. Происходит перемещение указателя к полю, расположенному ниже;

Блок 4. Стирается последний символ, введённый в текущее поле;

Блок 5. Вызывается функция отправки текущей формы;

Блок 6. Проверяется возможно ли добавление символа заданной клавиши;

Блок 7. Символ заданной клавиши добавляется в конец текущего поля.

* + 1. Описание алгоритма подпрограммы очистки формы ввода

Структурная схема алгоритма подпрограммы обработки нажатия клавиши формы ввода приведена на рисунке 2.7.



Рисунок 2.7 – Схема подпрограммы очистки формы ввода

В соответствии с алгоритмом подпрограмма функционирует следующим образом:

Блок 1. Происходит итерация по каждому полю заданной формы ввода;

Блок 2. Поле ввода заполняется символами «\0» и обнуляется индекс курсора.

* 1. **Выбор языка программирования**

При выборе языка программирования *C++* и среду *NetBeans* я руководствовался следующими критериями:

* Для разработки программ такого рода наиболее удобным является язык высокого уровня;
* Использование стандарта языка *C++* без всяких надстроек позволяет легко переносить и поддерживать исходный код программы;
* Скорость выполнения готовой программы на *C++* много выше скорости аналогичной программы на многих других языках;

Язык *C++* имеет огромное количество библиотек с разными функциями, которые упрощают написание программы. Сам язык очень гибок и широко используется, а объем создаваемого исполняемого файла достаточно мал. Достаточно удобная работа с указателями, что позволяет писать структуры данных на подобии двусвязного списка.

Поэтому в условиях данной задачи целесообразно выбрать язык *C++*, так как он наиболее точно подходит под условия поставленной задачи и позволяет легко реализовать интерфейс программы.

* 1. **Обоснование состава технических и программных средств**

Программа предназначена для работы в среде операционных систем семейства *Linux* или в их эмуляторах и занимает 150 килобайтов памяти.

Минимальные системные требования соответствуют системным требованиям самих операционных систем, т.к. программа не требовательна к вычислительной мощности компьютера и потребляет минимум оперативной памяти, требуемой на загрузку файла и хранения его в оперативной памяти, что значительно меньше требований самой операционной системы.

Программный продукт разрабатывался и тестировался на компьютере следующей конфигурации:

*CPU – Intel® Core™ i7-4710HQ (2,5GHz)*

*RAM – 12 GB*

*Video – NVIDIA GeForce GTX-860m*

*OS – Linux Mint 17.1 Rebecca*

Узел сборки *– GNU C++*

1. **ВЫПОЛНЕНИЕ ПРОГРАММЫ**

Для начала работы программы необходимо в консоли запустить исполняемый файл *cpp*. После запуска файла на экране пользователь может видеть интерфейс программы. Слева располагается меню программы и по ширине консоли расположена шапка таблицы, в которой будут отображаться данные. Это показано на рисунке 3.1.



Рисунок 3.1 – Интерфейс программы

Управление программой осуществляется путём нажатия клавиш перемещения курсоров, вверх и вниз, причем при каждом перемещении курсора, поочередно выделяется выбранным цветом каждый из пунктов меню. Для выбора нужного пункта меню необходимо нажать клавишу *Enter*. Для скрытия/показа меню программы следует нажать клавишу *TAB*. Подсказка этого действия находится внизу меню, когда оно открыто, и в левом верхнем углу, когда скрыто.

Для дальнейшей работы с программой необходимо создать базу данных. Выполнить данное действие можно двумя способами, которые мы рассмотрим ниже:

Способ 1. Для создания базы данных достаточно выбрать пункт меню «Добавление», переведя клавишами вверх-вниз курсор и нажать *Enter*. После чего появится окно организации двунаправленного списка с запросами на ввод необходимых данных. В случае попытки ввода некорректных данных или ввода большего количества символов, чем было зарезервировано, программа блокирует ввод.

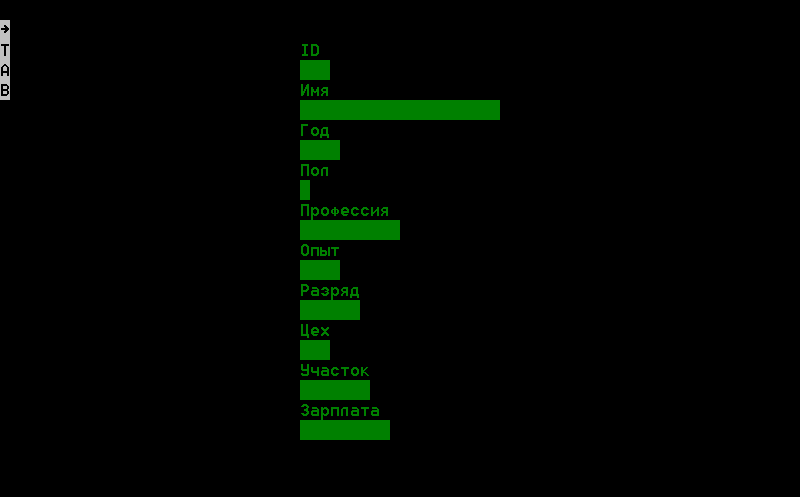


Рисунок 3.2 – Добавление элемента в список

Способ 2. Для создания базы данных достаточно загрузить данные из файла. Для этого необходимо выбрать пункт меню «Загрузка» данных и выбрать тип файла – бинарный или текстовый. Для корректной работы функции загрузки из файла рекомендуется вносить в программу данные, которые были записан непосредственно самой программой. Для того, чтобы создать такой файл, данные из программы необходимо сохранить в файл с указанием типа. Для этого необходимо ввести базу данных и вызвать «Сохранение» данных в файл из основного меню программы, путем выбора этого пункта. Так же есть возможность выхода из выбранного пункта главного меню путем выбора пункта «Назад».



Рисунок 3.3 – Сохранение данных в файл

В случае успешного завершения или же ошибки при считывании из файла, на экран выводится соответствующее сообщение.

После создания базы данных одним из выше приведенных способов, можно просмотреть ее, нажав *TAB* для скрытия меню. В результате пользователь видит окно с таблицей данных, в ней предусмотрена возможность прокрутки данных, для этого необходимо воспользоваться клавишами вверх-вниз.



Рисунок 3.4 – Просмотр данных

Для того, чтобы изменить данные надо выбрать необходимый элемент списка, открыть главное меню и выбрать пункт «Изменить». Далее появляется окно для редактирования, в котором пользователь вводит необходимые данные для изменения, если же пользователь не желает изменять какое-либо поле, то он, может пропустив его, нажав клавишу вниз. После нажатия *Enter* измененные данные сохранятся в таблицу. Выбрав необходимый номер, осуществляется поиск необходимого элемента и в случае его нахождения, появляется окно также предусмотрен выход из данной функции, он происходит путем выбора пункта «Назад» в открытом меню. После завершения редактирования, на экран выводиться таблица с измененными данными.

Если же пользователю нужно найти какой-либо элемент, он может это сделать, вызвав соответствующий пункт меню «Поиск». В результате появится меню поиска, содержащее поле ввода символов для поиска совпадений. Поиск возможен по названию товара или по дате продажи. В случае нахождения одной или нескольких записей, на экран выводится таблица результатов поиска, в которой видна вся информация найденных данных. Имеется возможность изменения и удаления выбранной записи. Если же совпадений не найдено, то выводится соответствующее сообщение.

Для того, чтобы удалить элемент из таблицы входных данных, необходимо в просмотре выбрать запись, которую пользователь хочет удалить, затем открыть меню и выбрать пункт «Удалить». В случае, если нужно удалить весь список, то это можно осуществить поэлементно с помощью этой же функции, или воспользоваться специально предназначенной для этого функцией «Очистка».

Помимо выше описанных функций, в программе реализованы и другие функции: Сортировка, Сохранение данных в файл, Обработка. Подробный механизм работы этих функций будет рассмотрен чуть позже.

1. **ТЕСТОВЫЙ ПРИМЕР РАБОТЫ ПРОГРАММЫ**

Как было сказано в предыдущем пункте, для работы с программой требуется загрузить базу данных из файла или создать новую, выбрав соответствующие пункты меню.

Загрузим данные из текстового файла, для этого выберем пункт меню «Загрузка» → «Из текстового». В случае успешного выполнения функции, на экране появится таблица данных.



Рисунок 4.1 – Просмотр базы данных

В просмотре реализован доступ к основным функциям программы. Проверим их работоспособность. Выберем первую запись и выберем пункт меню «Изменение».

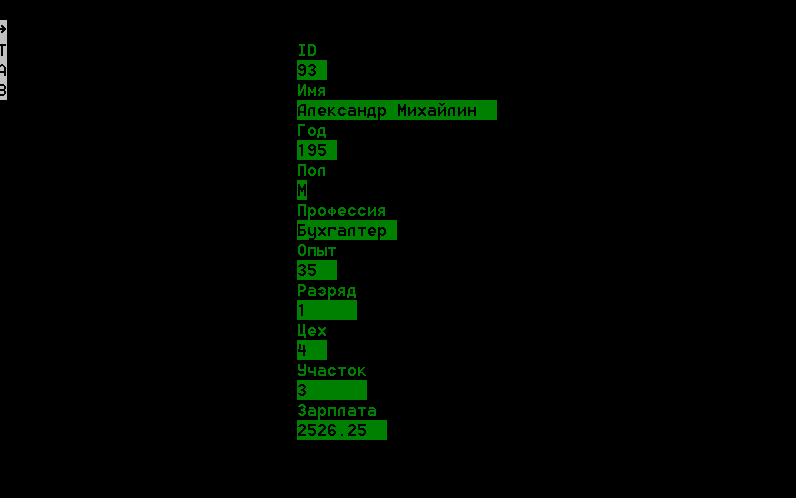


Рисунок 4.2 – Окно изменения данных записи

После нажатия *Enter* пользователь может видеть таблицу с измененными данными.



Рисунок 4.3 – Просмотр измененных данных

Выполним поиск по номеру цеха. Для этого выберем соответствующий пункт программы. Проведем поиск по букве «6».

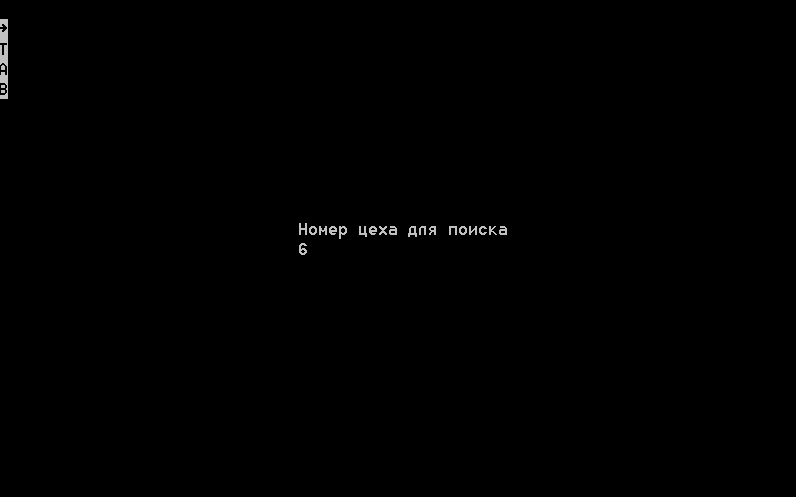


Рисунок 4.4 – Просмотр измененных данных

Программа нашла записи и вывела соответствующую таблицу.

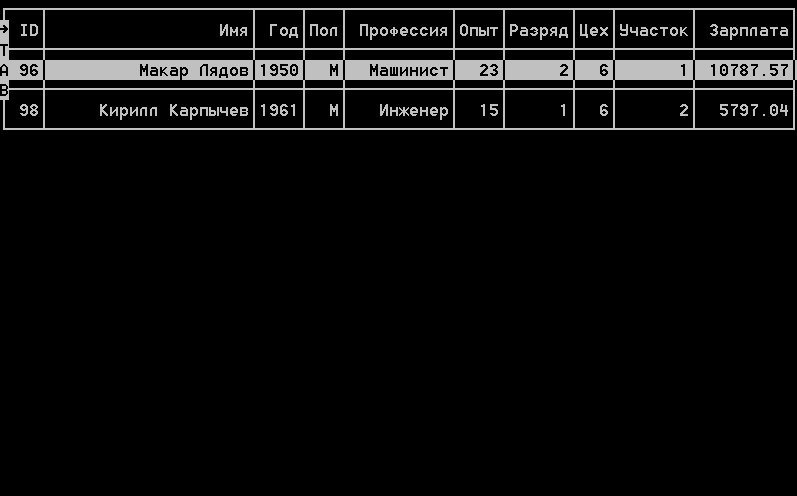


Рисунок 4.5 – Просмотр найденных записей

Удалим первую найденную запись, выбрав соответствующий пункт меню.

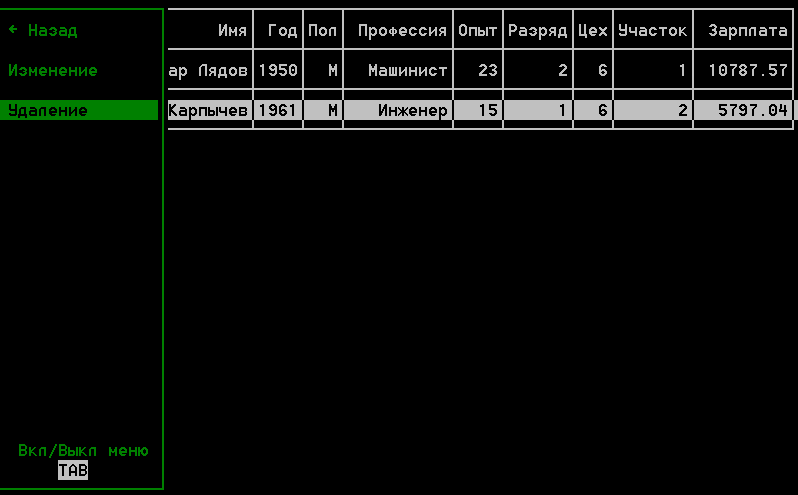


Рисунок 4.6 – Удаление одной из найденных записей

Оставшиеся данные обработаем, выбрав соответствующий пункт главного меню. Так как мы имеем мало данных, то анализируя сводную таблицу объема дневной выручки можем видеть, что каждый день совершалась покупка только одного вида товаров. В программе предусмотрена функция сохранения сводной таблицы для последующего использования пользователем или для печати. Для это следует с помощью *TAB* открыть меню и выбрать пункт «Сохранить». Данные сохранятся в текстовый файл.

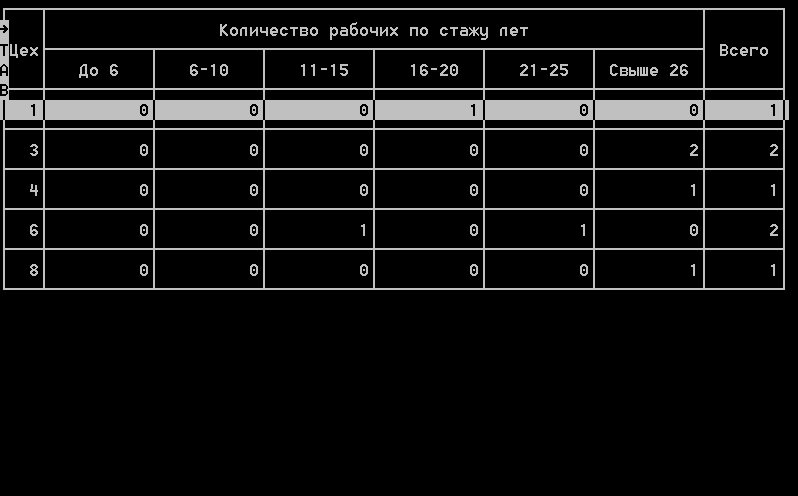


Рисунок 4.7 – Сводная таблица объема дневной выручки

Далее выйдем из сводной таблицы и в основном выбрать пункт «Сохранить» → «В текстовый».

После выйдем из программы и просмотрим сохраненные данные.

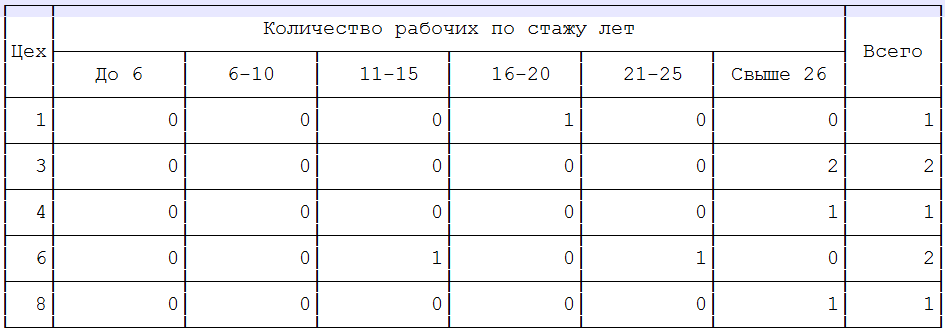


Рисунок 4.8 – Сохраненная сводная таблица

Рисунок 4.9 – Сохраненные данные

# **ЗАКЛЮЧЕНИЕ**

Программа, составлена в соответствии с постановкой задачи на курсовое проектирование по теме "Программа учета дневной выручки магазина". При написании программы использованы методические указания по курсовому проектированию по дисциплине "ОПиАЯ". Программа выполняет следующие функции: добавление, удаление, просмотр, сортировки по двум полям в восходящем и нисходящем порядке, редактирование, поиск, сохранения в файл, чтение из файла, сводную таблицу с подсчетом дневной выручки. Для проверки работоспособности программы и правильности обработки входных данных разработан тестовый пример. Тестирование программы подтвердило, что программа правильно выполняет обработку данных и выдаёт корректные результаты.

Программа предназначена для отдела торгового предприятия или магазина. Она удовлетворяет всем поставленным требованиям и пригодна для эксплуатации и к решению практических задач по обработке информации, ведения учёта за товарами и магазинами, а также их статистики.

В ходе разработки данного курсового проектирования были получены навыки создания программ с реальной целью, обработки ошибок пользователя и исключений. Также были закреплены знания по работе с информационными структурами, списками, указателями.

Таким образом, разработка данного проекта позволила закрепить и углубить знания о языке *С/С++* и разработке программ на основе концепции структурного программирования.

# **БИБЛИОГРАФИЧЕСКИЙ СПИСОК**

1. Страуструп Б. Язык программирования *C++* Специальное издание/ Б.Страуструп —Бином, 2011, 1136с.
2. Методические указания к курсовому проектированию по дисциплине «Основы программирования и алгоритмические языки». Сост. В.Н. Бондарев, В.Н. Хоролич. – Севастополь: Изд-во СевНТУ, 2008. – 98 с.
3. Голицына, О.Л. Языки программирования: учеб. пособие /О.Л. Голицына, Т.Л. Партыка, И.И. Попов. – М.: ФОРУМ, 2010. – 397 с.
4. Маслов, А.Н. Введение в язык программирования *C*: учеб.-справ. пособие /А.Н. Маслов. – М.: МИП «Память», 1991. – 62 с.
5. Подбельский В.В.Подбельский В.В. Программирование на языке СИ : [Учеб.пособие для студентов вузов] / В.В.Подбельский , Фомин С.С.. -2-е доп. изд. -М.: Финансы и статистика, 1999. -600 с

# **Приложение А**

**Текст программы**

/\*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Севастопольский государственный университет

Кафедра информационных систем

Программа подсчёта рабочих по стажу и цеху

Текст программы

RU.920101001.02463-01 81 12

РАЗРАБОТАЛ

Студент гр. ИС/б-24-о

Куркчи А.Э.

2015

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Программа подсчёта рабочих по стажу и цеху. В ней учитывается информация о рабочих: табельный номер, Ф.И.О. год рождения, пол, профессия, стаж работы, разряд, номер цеха, номер участка и зарплата. В результате работы программы выводится таблица, в которой подсчитывается количество рабочих по их стажу в каждом из цехов.

Вариант задания 12. Утверждено 14.09.2015

Среда программирования *NetBeans*

Дата последней коррекции: 16.12.15

Версия 1.0

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++\*/

1. **ФАЙЛ *MAIN.CPP***

#include <stdlib.h>

#include <string.h>

#include <locale.h>

#include <unistd.h>

#include <signal.h>

#include "list.h"

#include "display.h"

#include "form.h"

int main**(**int argc**,**char **\*\***argv**)** **{**

setlocale**(**LC\_ALL**,**""**);**

/\* Блокировка сигналов \*/

sigset\_t mask**;**

sigset\_t orig\_mask**;**

sigfillset**(&**mask**);**

sigaddset**(&**mask**,**SIGINT**);**

**if(**sigprocmask**(**SIG\_BLOCK**,&**mask**,&**orig\_mask**)** **<** 0**)** **{**

**return** **-**27**;**

**}**

/\* Генерация списка по файлу имён из первого параметра \*/

**if(**argc **>** 1**)** **{**

list**::**list\_t l **=** list**::**generate**(**argv**[**1**]);**

list**::**save**(**l**,**"data.bin"**,true);**

list**::**save**(**l**,**"data.txt"**,false);**

list**::**free**(**l**);**

**}**

/\* Инициализация форм \*/

form**::**init\_forms**();**

/\* Инициализация экрана \*/

display**::**init\_display**();**

/\* Отрисовка экрана \*/

display**::**draw**();**

/\* Делегация клавиш и отрисовка экрана \*/

**do** **{**

display**::**draw**();**

**}** **while(**display**::**key**());**

/\* Завершение рабоиы экрана \*/

display**::**close\_display**();**

**return** 0**;**

**}**

1. **ФАЙЛ *FORM.H***

#ifndef FORM\_H

#define FORM\_H

**namespace** form **{**

/\* Поле \*/

**typedef** struct field\_t **{**

/\* Имя \*/

wchar\_t**\*** name**;**

/\* Содержимое \*/

wchar\_t**\*** data**;**

/\* Индекс \*/

int index**;**

/\* Длина \*/

int length**;**

/\* Высота \*/

int height**;**

/\* Ширина \*/

int width**;**

/\* Y \*/

int y**;**

/\* X \*/

int x**;**

/\* Валидатор \*/

bool **(\***validator**)(**const wchar\_t**\*);**

**}** field\_t**;**

/\* Форма \*/

**typedef** struct form\_t **{**

/\* Поля \*/

field\_t**\*\*** field**;**

/\* Длина \*/

int length**;**

/\* Текущее поле\*/

int current\_field**;**

/\* Обработчик \*/

bool **(\***submit**)(**const form\_t**\*);**

**}** form\_t**;**

/\* Полная форма \*/

extern form\_t**\*** full\_form**;**

/\* Редактируемый элемент \*/

extern int current\_el**;**

/\* Инициализация форм \*/

void init\_forms**();**

/\* Создание нового поля \*/

field\_t**\*** new\_field**(**const wchar\_t**\*,**int**,**int**,**int**,**bool **(\*)(**const wchar\_t**\*));**

/\* Создание новой формы \*/

form\_t**\*** new\_form**(**field\_t**\*\*,**int**,**bool**(\*)(**const form\_t**\*));**

/\* Очистка полей формы \*/

void clear**(**form\_t**\*);**

/\* Обработчик клавиш \*/

bool key**(**form\_t**\*,**int c**);**

/\* Обработчик формы добавления \*/

bool add\_submit**(**const form\_t**\*);**

/\* Обработчик формы редактирования \*/

bool edit\_submit**(**const form\_t**\*);**

/\* Валидатор целого числа\*/

bool validator\_number**(**const wchar\_t**\*);**

/\* Валидатор слова \*/

bool validator\_word**(**const wchar\_t**\*);**

/\* Валидатор строки \*/

bool validator\_string**(**const wchar\_t**\*);**

/\* Валидатор денег\*/

bool validator\_money**(**const wchar\_t**\*);**

/\* Валидатор пола \*/

bool validator\_sex**(**const wchar\_t**\*);**

**}**

#endif /\* FORM\_H \*/

1. **ФАЙЛ *FORM.CPP***

#include <string.h>

#include <wchar.h>

#include "display.h"

#include "list.h"

#include "form.h"

**namespace** form **{**

form\_t**\*** full\_form **=** **NULL;**

int current\_el **=** 0**;**

void init\_forms**()** **{**

field\_t**\*\*** field **=** **new** field\_t**\*[**10**];**

field**[**0**]** **=** new\_field**(**L"ID"**,**3**,**2**,**30**,&**validator\_number**);**

field**[**1**]** **=** new\_field**(**L"Имя"**,**20**,**4**,**30**,&**validator\_string**);**

field**[**2**]** **=** new\_field**(**L"Год"**,**4**,**6**,**30**,&**validator\_number**);**

field**[**3**]** **=** new\_field**(**L"Пол"**,**1**,**8**,**30**,&**validator\_sex**);**

field**[**4**]** **=** new\_field**(**L"Профессия"**,**10**,**10**,**30**,&**validator\_string**);**

field**[**5**]** **=** new\_field**(**L"Опыт"**,**4**,**12**,**30**,&**validator\_number**);**

field**[**6**]** **=** new\_field**(**L"Разряд"**,**6**,**14**,**30**,&**validator\_number**);**

field**[**7**]** **=** new\_field**(**L"Цех"**,**3**,**16**,**30**,&**validator\_number**);**

field**[**8**]** **=** new\_field**(**L"Участок"**,**7**,**18**,**30**,&**validator\_number**);**

field**[**9**]** **=** new\_field**(**L"Зарплата"**,**9**,**20**,**30**,&**validator\_money**);**

full\_form **=** new\_form**(**field**,**10**,NULL);**

**}**

field\_t**\*** new\_field**(**const wchar\_t**\*** name**,**int len**,**int y**,**int x**,**bool **(\***validator**)(**const wchar\_t**\*))** **{**

field\_t**\*** r **=** **new** field\_t**;**

r**->**name **=** **new** wchar\_t**[**len**+**1**];**

wcscpy**(**r**->**name**,**name**);**

r**->**data **=** **new** wchar\_t**[**len**+**1**];**

r**->**data**[**0**]** **=** L'\0'**;**

r**->**index **=** 0**;**

r**->**length **=** len**;**

r**->**height **=** 2**;**

r**->**width **=** len**;**

r**->**y **=** y**;**

r**->**x **=** x**;**

r**->**validator **=** validator**;**

**return** r**;**

**}**

form\_t**\*** new\_form**(**field\_t**\*\*** field**,**int length**,**bool **(\***submit**)(**const form\_t**\*))** **{**

form\_t**\*** r **=** **new** form\_t**;**

r**->**field **=** field**;**

r**->**length **=** length**;**

r**->**current\_field **=** 0**;**

r**->**submit **=** submit**;**

**return** r**;**

**}**

void clear**(**form\_t**\*** f**)** **{**

**for(**int i**=**0**;**i**<**f**->**length**;**i**++)** **{**

f**->**field**[**i**]->**data**[**0**]** **=** L'\0'**;**

f**->**field**[**i**]->**index **=** 0**;**

**}**

f**->**current\_field **=** 0**;**

**}**

bool key**(**form\_t**\*** f**,**int c**)** **{**

**switch(**c**)** **{**

**case** KEY\_UP**:**

**if(**f**->**current\_field **==** 0**)** **{**

f**->**current\_field **=** f**->**length**-**1**;**

**}** **else** **{**

f**->**current\_field**--;**

**}**

**break;**

**case** KEY\_DOWN**:**

**if(**f**->**current\_field **==** f**->**length**-**1**)** **{**

f**->**current\_field **=** 0**;**

**}** **else** **{**

f**->**current\_field**++;**

**}**

**break;**

**case** KEY\_BACKSPACE**:**

**case** 127**:**

**if(**f**->**field**[**f**->**current\_field**]->**index **!=** 0**)** **{**

f**->**field**[**f**->**current\_field**]->**data**[--**f**->**field**[**f**->**current\_field**]->**index**]** **=** L'\0'**;**

**}**

**break;**

**break;**

**case** KEY\_ENTER**:**

**case** L'\n'**:**

f**->**submit**(**f**);**

**break;**

**default:**

**if(**c **&&** f**->**field**[**f**->**current\_field**]->**index **<** f**->**field**[**f**->**current\_field**]->**length**)** **{**

wchar\_t**\*** tmp **=** **new** wchar\_t**[**f**->**field**[**f**->**current\_field**]->**length**+**1**];**

wcscpy**(**tmp**,**f**->**field**[**f**->**current\_field**]->**data**);**

f**->**field**[**f**->**current\_field**]->**data**[**f**->**field**[**f**->**current\_field**]->**index**++]** **=** c**;**

f**->**field**[**f**->**current\_field**]->**data**[**f**->**field**[**f**->**current\_field**]->**index**]** **=** L'\0'**;**

**if(!**f**->**field**[**f**->**current\_field**]->**validator**(**f**->**field**[**f**->**current\_field**]->**data**))** **{**

wcscpy**(**f**->**field**[**f**->**current\_field**]->**data**,**tmp**);**

f**->**field**[**f**->**current\_field**]->**index**--;**

**}**

**delete** tmp**;**

**}**

**break;**

**}**

**return** **true;**

**}**

bool add\_submit**(**const form\_t**\*** f**)** **{**

**for(**int i**=**0**;**i**<**f**->**length**;**i**++)** **{**

**if(!**f**->**field**[**i**]->**index**)** **{**

**return** **false;**

**}**

**}**

list**::**data\_t d**;**

d**.**id **=** list**::**wtoi**(**f**->**field**[**0**]->**data**);**

wcscpy**(**d**.**name**,**f**->**field**[**1**]->**data**);**

d**.**b\_year **=** list**::**wtoi**(**f**->**field**[**2**]->**data**);**

d**.**sex **=** f**->**field**[**3**]->**data**[**0**];**

wcscpy**(**d**.**prof**,**f**->**field**[**4**]->**data**);**

d**.**exp **=** list**::**wtoi**(**f**->**field**[**5**]->**data**);**

d**.**rank **=** list**::**wtoi**(**f**->**field**[**6**]->**data**);**

d**.**guild **=** list**::**wtoi**(**f**->**field**[**7**]->**data**);**

d**.**sect **=** list**::**wtoi**(**f**->**field**[**8**]->**data**);**

d**.**salary **=** list**::**wtof**(**f**->**field**[**9**]->**data**);**

**if(!**list**::**push\_back**(**list**::**global**,**d**))** **{**

display**::**show\_message**(**display**::**message\_nomemory**);**

**}**

display**::**go\_home**();**

display**::**toggle\_menu**();**

**return** **true;**

**}**

bool edit\_submit**(**const form\_t**\*** f**)** **{**

**for(**int i**=**0**;**i**<**f**->**length**;**i**++)** **{**

**if(!**f**->**field**[**i**]->**index**)** **{**

**return** **false;**

**}**

**}**

list**::**data\_t d**;**

d**.**id **=** list**::**wtoi**(**f**->**field**[**0**]->**data**);**

wcscpy**(**d**.**name**,**f**->**field**[**1**]->**data**);**

d**.**b\_year **=** list**::**wtoi**(**f**->**field**[**2**]->**data**);**

d**.**sex **=** f**->**field**[**3**]->**data**[**0**];**

wcscpy**(**d**.**prof**,**f**->**field**[**4**]->**data**);**

d**.**exp **=** list**::**wtoi**(**f**->**field**[**5**]->**data**);**

d**.**rank **=** list**::**wtoi**(**f**->**field**[**6**]->**data**);**

d**.**guild **=** list**::**wtoi**(**f**->**field**[**7**]->**data**);**

d**.**sect **=** list**::**wtoi**(**f**->**field**[**8**]->**data**);**

d**.**salary **=** list**::**wtof**(**f**->**field**[**9**]->**data**);**

**if(!**list**::**update**(**list**::**global**,**form**::**current\_el**,**d**))** **{**

display**::**show\_message**(**display**::**message\_noelement**);**

**}**

display**::**go\_home**();**

display**::**toggle\_menu**();**

**return** **true;**

**}**

bool validator\_number**(**const wchar\_t**\*** s**)** **{**

**if(**s**[**0**]** **==** L'0'**)** **{**

**return** **false;**

**}**

int l **=** wcslen**(**s**);**

**for(**int i**=**0**;**i**<**l**;**i**++)** **{**

**if(!(**s**[**i**]** **>=** L'0' **&&** s**[**i**]** **<=** L'9'**))** **{**

**return** **false;**

**}**

**}**

**return** **true;**

**}**

bool validator\_word**(**const wchar\_t**\*** s**)** **{**

int l **=** wcslen**(**s**);**

**for(**int i**=**0**;**i**<**l**;**i**++)** **{**

**if(!((**s**[**i**]** **>=** L'0' **&&** s**[**i**]** **<=** L'9'**)** **||**

**(**s**[**i**]** **>=** L'a' **&&** s**[**i**]** **<=** L'z'**)** **||**

**(**s**[**i**]** **>=** L'A' **&&** s**[**i**]** **<=** L'Z'**)** **||**

**(**s**[**i**]** **>=** L'а' **&&** s**[**i**]** **<=** L'я'**)** **||**

**(**s**[**i**]** **>=** L'А' **&&** s**[**i**]** **<=** L'Я'**)** **||**

**(**s**[**i**]** **==** L'ё' **||** s**[**i**]** **==** L'Ё'**)** **||**

**(**s**[**i**]** **==** L'-' **||** s**[**i**]** **==** L'\_'**)**

**))** **{**

**return** **false;**

**}**

**}**

**return** **true;**

**}**

bool validator\_string**(**const wchar\_t**\*** s**)** **{**

int l **=** wcslen**(**s**);**

**for(**int i**=**0**;**i**<**l**;**i**++)** **{**

**if(!((**s**[**i**]** **>=** L'0' **&&** s**[**i**]** **<=** L'9'**)** **||**

**(**s**[**i**]** **>=** L'a' **&&** s**[**i**]** **<=** L'z'**)** **||**

**(**s**[**i**]** **>=** L'A' **&&** s**[**i**]** **<=** L'Z'**)** **||**

**(**s**[**i**]** **>=** L'а' **&&** s**[**i**]** **<=** L'я'**)** **||**

**(**s**[**i**]** **>=** L'А' **&&** s**[**i**]** **<=** L'Я'**)** **||**

**(**s**[**i**]** **==** L'ё' **||** s**[**i**]** **==** L'Ё'**)** **||**

**(**s**[**i**]** **==** L' ' **||** s**[**i**]** **==** L'-' **||** s**[**i**]** **==** L'\_'**)**

**))** **{**

**return** **false;**

**}**

**}**

**return** **true;**

**}**

bool validator\_money**(**const wchar\_t**\*** s**)** **{**

**if(**s**[**0**]** **==** L'0'**)** **{**

**return** **false;**

**}**

bool dot **=** **false;**

int l **=** wcslen**(**s**);**

**for(**int i**=**0**;**i**<**l**;**i**++)** **{**

**if((**s**[**i**]** **>=** L'0' **&&** s**[**i**]** **<=** L'9'**)** **||** s**[**i**]** **==** L'.' **||** s**[**i**]** **==** L','**)** **{**

**if(**s**[**i**]** **==** L'.' **||** s**[**i**]** **==** L','**)** **{**

**if(!**dot**)** **{**

**if(**i **==** 0**)** **{**

**return** **false;**

**}** **else** **{**

dot **=** **true;**

**}**

**}** **else** **{**

**return** **false;**

**}**

**}**

**}** **else** **{**

**return** **false;**

**}**

**}**

**return** **true;**

**}**

bool validator\_sex**(**const wchar\_t**\*** s**)** **{**

int l **=** wcslen**(**s**);**

**for(**int i**=**0**;**i**<**l**;**i**++)** **{**

**if(!(**s**[**i**]** **==** L'М' **||** s**[**i**]** **==** L'Ж'**))** **{**

**return** **false;**

**}**

**}**

**return** **true;**

**}**

**}**

1. **ФАЙЛ *LIST.H***

#ifndef LIST\_H

#define LIST\_H

#define DATA\_NAME\_LEN 20

#define DATA\_PROF\_LEN 10

**namespace** list **{**

/\* Элемент сводной таблицы \*/

**typedef** struct result\_t **{**

/\* Номер цеха \*/

int guild**;**

/\* До 6 \*/

int cat1**;**

/\* 6-10 \*/

int cat2**;**

/\* 11-15\*/

int cat3**;**

/\* 16-20 \*/

int cat4**;**

/\* 21-25 \*/

int cat5**;**

/\* Свыше 25 \*/

int cat6**;**

/\* Всего \*/

int total**;**

**}** result\_t**;**

/\* Блок данных \*/

**typedef** struct data\_t **{**

/\* Табельный номер \*/

int id**;**

/\* ФИО \*/

wchar\_t name **[**DATA\_NAME\_LEN**+**1**];**

/\* Год рождения \*/

int b\_year**;**

/\* Пол \*/

wchar\_t sex**;**

/\* Профессия \*/

wchar\_t prof **[**DATA\_PROF\_LEN**+**1**];**

/\* Стаж/опыт работы \*/

int exp**;**

/\* Разряд \*/

int rank**;**

/\* Номер цеха \*/

int guild**;**

/\* Номер участка \*/

int sect**;**

/\* Зарплата \*/

float salary**;**

**}** data\_t**;**

/\* Элемент списка \*/

**typedef** struct node\_t **{**

/\* Индекс \*/

int index**;**

/\* Данные \*/

data\_t data**;**

/\* Указатель на предыдущий элемент \*/

node\_t**\*** prev**;**

/\* Указатель на следующий элемент \*/

node\_t**\*** next**;**

**}** node\_t**;**

/\* Список \*/

**typedef** struct list\_t **{**

/\* Голова списка \*/

node\_t**\*** head**;**

/\* Хвост списка \*/

node\_t**\*** tail**;**

/\* Количество элементов \*/

int length**;**

**}** list\_t**;**

extern list\_t global**;**

extern list\_t**\*** current**;**

/\*

\* Создаёт пустой список

\*

\* @return list\_t созданный список

\*/

list\_t create**();**

/\*

\* Добавляет элемент в конец списка

\*

\* @param list\_t& список в который добавлять

\* @param const data\_t& данные которые добавлять

\*

\* @return bool успешно ли добавилось

\*/

bool push\_back**(**list\_t**&,**const data\_t**&);**

/\*

\* Изменяет данные в элементе списка

\*

\* @param list\_t& список в котором изменять

\* @param const int индекс по которому изменять

\* @param const data\_t& данные для изменения

\*

\* @return bool успешно ли изменилось

\*/

bool update**(**list\_t**&,**const int**,**const data\_t**&);**

/\*

\* Удаляет элемент списка

\*

\* @param list\_t& список в котором удалять

\* @param const int индекс который удалять

\*

\* @return bool успешно ли удалилось

\*/

bool erase**(**list\_t**&,**const int**);**

/\*

\* Возвращает элемент списка

\*

\* @param list\_t& список в котором найти

\* @param const int индекс который найти

\*

\* @return node\_t\* элемент по индексу (или NULL если его нет)

\*/

node\_t**\*** get**(**list\_t**&,**const int**);**

/\*

\* Ищет элемент в списке

\*

\* @param list\_t& список в котором найти

\* @param const data\_t& данные с которыми сравнивать

\*

\* @return list\_t\* найденные элементы

\*/

list\_t**\*** find**(**list\_t**&,**const data\_t**&);**

/\*

\* Сортирует список

\*

\* @param list\_t& список который сортировать

\* @param bool (\*)(data\_t&,data\_t&) компаратор

\*

\* @return void

\*/

void sort**(**list\_t**&,**bool **(\*)(**data\_t**&,**data\_t**&),**bool **=** **false);**

/\*

\* Компаратор по табельному номеру

\*

\* @param data\_t& первый элемент

\* @param data\_t& второй элемент

\*

\* @return bool результат сравнения

\*/

bool compare\_by\_id**(**data\_t**&,**data\_t**&);**

/\*

\* Компаратор по имени

\*

\* @param data\_t& первый элемент

\* @param data\_t& второй элемент

\*

\* @return bool результат сравнения

\*/

bool compare\_by\_name**(**data\_t**&,**data\_t**&);**

/\*

\* Компаратор по професси

\*

\* @param data\_t& первый элемент

\* @param data\_t& второй элемент

\*

\* @return bool результат сравнения

\*/

bool compare\_by\_prof**(**data\_t**&,**data\_t**&);**

/\*

\* Компаратор по зарплате

\*

\* @param data\_t& первый элемент

\* @param data\_t& второй элемент

\*

\* @return bool результат сравнения

\*/

bool compare\_by\_salary**(**data\_t**&,**data\_t**&);**

/\*

\* Считывает список из файла

\*

\* @param list\_t& список в который считывать данные

\* @param const char\* имя файла

\* @param bool бинарный ли файл

\*

\* @return bool успешно ли считывание

\*/

bool read**(**list\_t**&,**const char**\*,**bool**);**

/\*

\* Сохраняет список в файл

\*

\* @param list\_t& список который сохранять

\* @param const char\* имя файла

\* @param bool бинарный ли файл

\*

\* @return bool успешна ли запись

\*/

bool save**(**list\_t**&,**const char**\*,**bool**);**

/\*

\* Сохраняет сводную таблицу в файл

\*

\* @param result\_t\* сводная таблица которую сохранять

\* @param const int длина сводной таблицы

\* @param const char\* имя файла

\*

\* @return bool успешна ли запись

\*/

bool save**(**result\_t**\*,**const int**,**const char**\*);**

/\*

\* Глубоко копирует элемент

\*

\* @param const data\_t& данные которые копировать

\*

\* @return data\_t скопированные данные

\*/

data\_t copy**(**const data\_t**&);**

/\*

\* Сравнивает элементы по всем полям

\*

\* @param const data\_t& первый элемент

\* @param const data\_t& второй элемент

\*

\* @return bool похожи ли элементы

\*/

bool compare**(**const data\_t**&,**const data\_t**&);**

/\*

\* Освобождает память от списка

\*

\* @param const list\_t& список для очистки

\*

\* @return void

\*/

void free**(**list\_t**&);**

/\*

\* Создаёт сводную таблицу

\*

\* @param list\_t& список для обработки

\* @param int& длина сводной таблицы

\*

\* @return result\_t\* сводная таблица

\*/

result\_t**\*** process**(**list\_t**&,**int**&);**

/\*

\* Генерирует список из имён

\*

\* @param const char\* имя файла с именами

\*

\* @return list\_t созданный файл

\*/

list\_t generate**(**const char**\*);**

/\*

\* Удаляет лидирующие пробелы

\*

\* @param wchar\_t\* строка которую очистить от пробелов

\*

\* @return void

\*/

void erase\_spaces**(**wchar\_t**\*);**

/\*

\* Преобразует строку в целое число

\*

\* @param const wchar\_t\* числовая строка

\*

\* @return int число

\*/

int wtoi**(**const wchar\_t**\*);**

/\*

\* Преобразует целое число в строку

\*

\* @param int число

\* @param int максимальная длина

\*

\* @return const wchar\_t\* числовая строка

\*/

wchar\_t**\*** itow**(**const int**,**const int**);**

/\*

\* Преобразует строку в число с плавающей точкой

\*

\* @param const wchar\_t\* числовая строка

\*

\* @return число с плавающей точкой

\*/

float wtof**(**const wchar\_t**\*);**

/\*

\* Преобразует число с плавающей точкой в строку

\*

\* @param const wchar\_t\* формат

\* @param float число

\* @param int максимальная длина

\*

\* @return const wchar\_t\* числовая строка

\*/

wchar\_t**\*** ftow**(**const wchar\_t**\*,**const float**,**const int**);**

**}**

#endif /\* LIST\_H \*/

1. **ФАЙЛ *LIST.CPP***

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <wchar.h>

#define min(A,B) (((A)>(B))?(B):(A))

#include "list.h"

**namespace** list **{**

list\_t global **=** create**();**

list\_t**\*** current **=** **&**global**;**

list\_t create**()** **{**

list\_t l**;**

l**.**head **=** l**.**tail **=** **NULL;**

l**.**length **=** 0**;**

**return** l**;**

**}**

bool push\_back**(**list\_t**&** l**,**const data\_t**&** d**)** **{**

node\_t**\*** n **=** **new** node\_t**;**

**if(**n **==** **NULL)** **{**

**return** **false;**

**}**

n**->**index **=** l**.**length**;**

n**->**data **=** copy**(**d**);**

n**->**prev **=** l**.**tail**;**

n**->**next **=** **NULL;**

**if(**l**.**head **==** **NULL)** **{**

l**.**head **=** n**;**

**}**

**if(**l**.**tail **!=** **NULL)** **{**

l**.**tail**->**next **=** n**;**

**}**

l**.**tail **=** n**;**

l**.**length**++;**

**return** **true;**

**}**

bool update**(**list\_t**&** l**,**const int i**,**const data\_t**&** d**)** **{**

node\_t**\*** n **=** get**(**l**,**i**);**

**if(**n **==** **NULL)** **{**

**return** **false;**

**}**

n**->**data **=** copy**(**d**);**

**return** **true;**

**}**

bool erase**(**list\_t**&** l**,**const int i**)** **{**

node\_t**\*** n **=** get**(**l**,**i**);**

**if(**n **==** **NULL)** **{**

**return** **false;**

**}**

**if(**n**->**prev **==** **NULL)** **{**

l**.**head **=** n**->**next**;**

**}** **else** **{**

n**->**prev**->**next **=** n**->**next**;**

**}**

**if(**n**->**next **==** **NULL)** **{**

l**.**tail **=** **NULL;**

**}** **else** **{**

n**->**next**->**prev **=** n**->**prev**;**

node\_t**\*** tmp **=** n**->**next**;**

**while(**tmp **!=** **NULL)** **{**

tmp**->**index**--;**

tmp **=** tmp**->**next**;**

**}**

**}**

l**.**length**--;**

**delete** n**;**

**return** **true;**

**}**

node\_t**\*** get**(**list\_t**&** l**,**const int i**)** **{**

**if(**l**.**length **<=** i**)** **{**

**return** **NULL;**

**}**

int j **=** 0**;**

node\_t**\*** ptr **=** l**.**head**;**

**while(**ptr **!=** **NULL** **&&** j **<** i**)** **{**

ptr **=** ptr**->**next**;**

j**++;**

**}**

**return** ptr**;**

**}**

list\_t**\*** find**(**list\_t**&** l**,**const data\_t**&** d**)** **{**

list\_t**\*** found **=** **new** list\_t**;**

**\***found **=** create**();**

node\_t**\*** ptr **=** l**.**head**;**

**while(**ptr **!=** **NULL)** **{**

**if(**compare**(**ptr**->**data**,**d**))** **{**

push\_back**(\***found**,**ptr**->**data**);**

**}**

ptr **=** ptr**->**next**;**

**}**

**return** found**;**

**}**

void sort**(**list\_t**&** l**,**bool **(\***f**)(**data\_t**&,**data\_t**&),**bool r**)** **{**

**if(**l**.**head **!=** **NULL)** **{**

node\_t**\*** i **=** l**.**head**;**

node\_t**\*** j**;**

node\_t**\*** m**;**

**while(**i**->**next **!=** **NULL)** **{**

m **=** i**;**

j **=** i**->**next**;**

**while(**j **!=** **NULL)** **{**

bool cmp **=** f**(**m**->**data**,**j**->**data**);**

cmp **=** **(**r **&&** **!**cmp**)** **||** **(!**r **&&** cmp**);**

**if(**cmp**)** **{**

m **=** j**;**

**}**

j **=** j**->**next**;**

**}**

**if(**i **!=** m**)** **{**

data\_t tmp **=** i**->**data**;**

i**->**data **=** copy**(**m**->**data**);**

m**->**data **=** copy**(**tmp**);**

**}**

i **=** i**->**next**;**

**}**

**}**

**}**

bool compare\_by\_id**(**data\_t**&** a**,**data\_t**&** b**)** **{**

**return** a**.**id **>** b**.**id**;**

**}**

bool compare\_by\_name**(**data\_t**&** a**,**data\_t**&** b**)** **{**

**return** wcscmp**(**a**.**name**,**b**.**name**)** **>** 0**;**

**}**

bool compare\_by\_prof**(**data\_t**&** a**,**data\_t**&** b**)** **{**

**return** wcscmp**(**a**.**prof**,**b**.**prof**)** **>** 0**;**

**}**

bool compare\_by\_salary**(**data\_t**&** a**,**data\_t**&** b**)** **{**

**return** a**.**salary **>** b**.**salary**;**

**}**

bool read**(**list\_t**&** l**,**const char**\*** filename**,**bool binary**)** **{**

FILE**\*** f **=** fopen**(**filename**,**binary**?**"rb"**:**"r"**);**

**if(**f **==** **NULL)** **{**

**return** **NULL;**

**}**

data\_t data**;**

**if(**binary**)** **{**

**while(**fread**(&**data**,sizeof(**data\_t**),**1**,**f**)** **>** 0**)** **{**

push\_back**(**l**,**data**);**

**}**

**}** **else** **{**

wchar\_t trash**[**100**];**

fgetws**(**trash**,**100**,**f**);** // skip line

fgetws**(**trash**,**100**,**f**);** // 3

fgetws**(**trash**,**100**,**f**);** // times

**while(**fgetwc**(**f**)** **!=** WEOF**)** **{**

fwscanf**(**f**,**L"%d"**,&**data**.**id**);**

fgetwc**(**f**);**

fgetws**(**data**.**name**,**DATA\_NAME\_LEN**+**1**,**f**);**

erase\_spaces**(**data**.**name**);**

fgetwc**(**f**);**

fwscanf**(**f**,**L"%d"**,&**data**.**b\_year**);**

fgetwc**(**f**);**

fgetwc**(**f**);** // 2 spaces

fgetwc**(**f**);** // 2 spaces

wchar\_t sex **=** fgetwc**(**f**);**

data**.**sex **=** sex **==** L'М' **?** L'М' **:** L'Ж'**;**

fgetwc**(**f**);**

fgetws**(**data**.**prof**,**DATA\_PROF\_LEN**+**1**,**f**);**

erase\_spaces**(**data**.**prof**);**

fgetwc**(**f**);**

fwscanf**(**f**,**L"%d"**,&**data**.**exp**);**

fgetwc**(**f**);**

fwscanf**(**f**,**L"%d"**,&**data**.**rank**);**

fgetwc**(**f**);**

fwscanf**(**f**,**L"%d"**,&**data**.**guild**);**

fgetwc**(**f**);**

fwscanf**(**f**,**L"%d"**,&**data**.**sect**);**

fgetwc**(**f**);**

fwscanf**(**f**,**L"%f"**,&**data**.**salary**);**

fgetwc**(**f**);**

fgetwc**(**f**);**

fgetws**(**trash**,**100**,**f**);** // skip line

push\_back**(**l**,**data**);**

**}**

**}**

fclose**(**f**);**

**return** **true;**

**}**

bool save**(**list\_t**&** l**,**const char**\*** filename**,**bool binary**)** **{**

FILE**\*** f **=** fopen**(**filename**,**binary**?**"wb"**:**"w"**);**

**if(**f **==** **NULL)** **{**

**return** **false;**

**}**

**if(!**binary**)** **{**

fwprintf**(**f**,**L"┌───┬────────────────────┬────┬───┬──────────┬────┬──────┬───┬───────┬─────────┐\n"**);**

fwprintf**(**f**,**L"│%3S│%20S│%4S│%3S│%10S│%4S│%6S│%3S│%7S│%9S│\n"**,**

L"ID"**,**

L"Имя"**,**

L"Год"**,**

L"Пол"**,**

L"Профессия"**,**

L"Опыт"**,**

L"Разряд"**,**

L"Цех"**,**

L"Участок"**,**

L"Зарплата"**);**

**}**

node\_t**\*** ptr **=** l**.**head**;**

**while(**ptr **!=** **NULL)** **{**

**if(**binary**)** **{**

fwrite**(&(**ptr**->**data**),sizeof(**data\_t**),**1**,**f**);**

**}** **else** **{**

fwprintf**(**f**,**L"├───┼────────────────────┼────┼───┼──────────┼────┼──────┼───┼───────┼─────────┤\n"**);**

fwprintf**(**f**,**L"│%3d│%20S│%4d│%3C│%10S│%4d│%6d│%3d│%7d│%9.2f│\n"**,**

ptr**->**data**.**id**,**

ptr**->**data**.**name**,**

ptr**->**data**.**b\_year**,**

ptr**->**data**.**sex**,**

ptr**->**data**.**prof**,**

ptr**->**data**.**exp**,**

ptr**->**data**.**rank**,**

ptr**->**data**.**guild**,**

ptr**->**data**.**sect**,**

ptr**->**data**.**salary**);**

**}**

ptr **=** ptr**->**next**;**

**}**

**if(!**binary**)** **{**

fwprintf**(**f**,**L"└───┴────────────────────┴────┴───┴──────────┴────┴──────┴───┴───────┴─────────┘\n"**);**

**}**

fclose**(**f**);**

**return** **true;**

**}**

bool save**(**result\_t**\*** r**,**const int n**,**const char**\*** filename**)** **{**

FILE**\*** f **=** fopen**(**filename**,**"w"**);**

**if(**f **==** **NULL)** **{**

**return** **false;**

**}**

fwprintf**(**f**,**L"┌───┬─────────────────────────────────────────────────────────────────┬───────┐\n"**);**

fwprintf**(**f**,**L"│ │ Количество рабочих по стажу лет │ │\n"**);**

fwprintf**(**f**,**L"│Цех├──────────┬──────────┬──────────┬──────────┬──────────┬──────────┤ Всего │\n"**);**

fwprintf**(**f**,**L"│ │ До 6 │ 6-10 │ 11-15 │ 16-20 │ 21-25 │ Свыше 26 │ │\n"**);**

**for(**int i**=**0**;**i**<**n**;**i**++)** **{**

**if(**r**[**i**].**total**)** **{**

fwprintf**(**f**,**L"├───┼──────────┼──────────┼──────────┼──────────┼──────────┼──────────┼───────┤\n"**);**

fwprintf**(**f**,**L"│%3d│%10d│%10d│%10d│%10d│%10d│%10d│%7d│\n"**,**

r**[**i**].**guild**,**

r**[**i**].**cat1**,**

r**[**i**].**cat2**,**

r**[**i**].**cat3**,**

r**[**i**].**cat4**,**

r**[**i**].**cat5**,**

r**[**i**].**cat6**,**

r**[**i**].**total**);**

**}**

**}**

fwprintf**(**f**,**L"└───┴──────────┴──────────┴──────────┴──────────┴──────────┴──────────┴───────┘\n"**);**

fclose**(**f**);**

**return** **true;**

**}**

data\_t copy**(**const data\_t**&** d**)** **{**

data\_t n**;**

n **=** d**;**

wcscpy**(**n**.**name**,**d**.**name**);**

wcscpy**(**n**.**prof**,**d**.**prof**);**

**return** n**;**

**}**

bool compare**(**const data\_t**&** f**,**const data\_t**&** s**)** **{**

**if(**s**.**id **!=** **-**1 **&&** f**.**id **!=** s**.**id**)** **{**

**return** **false;**

**}**

**if(**wcslen**(**s**.**name**)** **!=** 0 **&&** wcsstr**(**f**.**name**,**s**.**name**)** **==** **NULL)** **{**

**return** **false;**

**}**

**if(**s**.**b\_year **!=** **-**1 **&&** f**.**b\_year **!=** s**.**b\_year**)** **{**

**return** **false;**

**}**

**if(**s**.**sex **!=** L'\0' **&&** f**.**sex **!=** s**.**sex**)** **{**

**return** **false;**

**}**

**if(**wcslen**(**s**.**prof**)** **!=** 0 **&&** wcsstr**(**f**.**prof**,**s**.**prof**)** **==** **NULL)** **{**

**return** **false;**

**}**

**if(**s**.**exp **!=** **-**1 **&&** f**.**exp **!=** s**.**exp**)** **{**

**return** **false;**

**}**

**if(**s**.**rank **!=** **-**1 **&&** f**.**rank **!=** s**.**rank**)** **{**

**return** **false;**

**}**

**if(**s**.**guild **!=** **-**1 **&&** f**.**guild **!=** s**.**guild**)** **{**

**return** **false;**

**}**

**if(**s**.**sect **!=** **-**1 **&&** f**.**sect **!=** s**.**sect**)** **{**

**return** **false;**

**}**

**if(**s**.**salary **!=** **-**1 **&&** f**.**salary **!=** s**.**salary**)** **{**

**return** **false;**

**}**

**return** **true;**

**}**

void free**(**list\_t**&** l**)** **{**

node\_t**\*** ptr **=** l**.**head**;**

node\_t**\*** tmp**;**

**while(**ptr **!=** **NULL)** **{**

tmp **=** ptr**->**next**;**

**delete** ptr**;**

ptr **=** tmp**;**

**}**

l**.**head **=** l**.**tail **=** **NULL;**

l**.**length **=** 0**;**

**}**

result\_t**\*** process**(**list\_t**&** l**,**int**&** n**)** **{**

result\_t**\*** r **=** **new** result\_t**[**l**.**length**];**

n **=** 0**;**

int k**;**

node\_t**\*** tmp **=** l**.**head**;**

**while(**tmp **!=** **NULL)** **{**

k **=** n**;**

**for(**int i**=**0**;**i**<**n**;**i**++)** **{**

**if(**r**[**i**].**guild **==** tmp**->**data**.**guild**)** **{**

k **=** i**;**

**break;**

**}**

**}**

**if(**k **==** n**)** **{**

r**[**k**].**guild **=** tmp**->**data**.**guild**;**

r**[**k**].**cat1 **=** 0**;**

r**[**k**].**cat2 **=** 0**;**

r**[**k**].**cat3 **=** 0**;**

r**[**k**].**cat4 **=** 0**;**

r**[**k**].**cat5 **=** 0**;**

r**[**k**].**cat6 **=** 0**;**

r**[**k**].**total **=** 0**;**

n**++;**

**}**

r**[**k**].**total**++;**

**if(**tmp**->**data**.**exp **<** 6**)** **{**

r**[**k**].**cat1**++;**

**}** **else** **if(**tmp**->**data**.**exp **>=** 6 **&&** tmp**->**data**.**exp **<=** 10**)** **{**

r**[**k**].**cat2**++;**

**}** **else** **if(**tmp**->**data**.**exp **>=** 11 **&&** tmp**->**data**.**exp **<=** 15**)** **{**

r**[**k**].**cat3**++;**

**}** **else** **if(**tmp**->**data**.**exp **>=** 16 **&&** tmp**->**data**.**exp **<=** 20**)** **{**

r**[**k**].**cat4**++;**

**}** **else** **if(**tmp**->**data**.**exp **>=** 21 **&&** tmp**->**data**.**exp **<=** 25**)** **{**

r**[**k**].**cat5**++;**

**}** **else** **if(**tmp**->**data**.**exp **>=** 26**)** **{**

r**[**k**].**cat6**++;**

**}**

tmp **=** tmp**->**next**;**

**}**

int m**;**

result\_t t**;**

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

m **=** i**;**

**for(**int j**=**i**+**1**;**j**<**n**;**j**++)** **{**

**if(**r**[**j**].**guild **<** r**[**m**].**guild**)** **{**

m **=** j**;**

**}**

**}**

**if(**m **!=** i**)** **{**

t **=** r**[**m**];**

r**[**m**]** **=** r**[**i**];**

r**[**i**]** **=** t**;**

**}**

**}**

/\*

fwprintf(f,L"┌───┬─────────────────────────────────────────────────────────────────┬───────┐\n");

fwprintf(f,L"│ │ Количество рабочих по стажу лет │ │\n");

fwprintf(f,L"│Цех├──────────┬──────────┬──────────┬──────────┬──────────┬──────────┤ Всего │\n");

fwprintf(f,L"│ │ До 6 │ 6-10 │ 11-15 │ 16-20 │ 21-25 │ Свыше 26 │ │\n");

for(int i=0;i<1000;i++) {

if(r[i].total) {

fwprintf(f,L"├───┼──────────┼──────────┼──────────┼──────────┼──────────┼──────────┼───────┤\n");

fwprintf(f,L"│%3d│%10d│%10d│%10d│%10d│%10d│%10d│%7d│\n",

i,

r[i].cat1,

r[i].cat2,

r[i].cat3,

r[i].cat4,

r[i].cat5,

r[i].cat6,

r[i].total);

}

}

fwprintf(f,L"└───┴──────────┴──────────┴──────────┴──────────┴──────────┴──────────┴───────┘\n");

fclose(f);

\*/

**return** r**;**

**}**

list\_t generate**(**const char**\*** filename**)** **{**

static const wchar\_t**\*** profs**[]** **=** **{**

L"Инженер"**,**

L"Плотник"**,**

L"Слесарь"**,**

L"Машинист"**,**

L"Бухгалтер"

**};**

list\_t l **=** create**();**

FILE**\*** f **=** fopen**(**filename**,**"r"**);**

**if(**f **==** **NULL)** **{**

wprintf**(**L"Файл не существует\n"**);**

**return** l**;**

**}**

wchar\_t sex**;**

wchar\_t name**[**DATA\_NAME\_LEN**+**1**];**

int i **=** 0**;**

**while(**fwscanf**(**f**,**L" %C"**,&**sex**)** **!=** EOF**)** **{**

fgetws**(**name**,**DATA\_NAME\_LEN**+**1**,**f**);**

name**[**wcslen**(**name**)-**2**]** **=** '\0'**;**

data\_t data**;**

data**.**id **=** **++**i**;**

wcscpy**(**data**.**name**,**name**);**

data**.**b\_year **=** 1950 **+** rand**()%**40**;**

data**.**sex **=** sex **==** L'М'**;**

wcscpy**(**data**.**prof**,**profs**[**rand**()%**5**]);**

data**.**exp **=** 1**+**rand**()%**35**;**

data**.**rank **=** 1**+**rand**()%**3**;**

data**.**guild **=** 1**+**rand**()%**10**;**

data**.**sect **=** 1**+**rand**()%**3**;**

data**.**salary **=** **(**150000**+(**rand**()%**1000000**))/**100.0**;**

push\_back**(**l**,**data**);**

**}**

fclose**(**f**);**

**return** l**;**

**}**

void erase\_spaces**(**wchar\_t**\*** str**)** **{**

int i**;**

int len **=** wcslen**(**str**);**

**for(**i**=**0**;**i**<**len **&&** str**[**i**]** **==** L' '**;**i**++);**

int k**=**0**;**

**for(**int j**=**i**;**j**<**len**;**j**++)** **{**

str**[**k**++]** **=** str**[**j**];**

**}**

str**[**k**]** **=** L'\0'**;**

**}**

int wtoi**(**const wchar\_t**\*** s**)** **{**

int len **=** wcslen**(**s**);**

int ret **=** 0**;**

**for(**int i**=**0**;**i**<**len**;**i**++)** **{**

ret **=** ret**\***10 **+** **(**s**[**i**]-**L'0'**);**

**}**

**return** ret**;**

**}**

wchar\_t**\*** itow**(**const int n**,**const int len**)** **{**

wchar\_t**\*** s **=** **new** wchar\_t**[**len**];**

swprintf**(**s**,**len**,**L"%d"**,**n**);**

**return** s**;**

**}**

float wtof**(**const wchar\_t**\*** s**)** **{**

int len **=** wcslen**(**s**);**

float ret **=** 0**;**

float exp **=** 0**;**

**for(**int i**=**0**;**i**<**len**;**i**++)** **{**

**if(**s**[**i**]** **==** L'.' **||** s**[**i**]** **==** L','**)** **{**

exp **=** 1**;**

**continue;**

**}**

exp **/=** 10.0**;**

ret **=** ret**\***10 **+** **(**s**[**i**]-**L'0'**);**

**}**

**return** ret**\***exp**;**

**}**

wchar\_t**\*** ftow**(**const wchar\_t**\*** format**,**const float n**,**const int len**)** **{**

wchar\_t**\*** s **=** **new** wchar\_t**[**len**];**

swprintf**(**s**,**len**,**format**,**n**);**

**return** s**;**

**}**

**}**

1. **ФАЙЛ *DISPLAY.H***

#ifndef DISPLAY\_H

#define DISPLAY\_H

#define NCURSES\_WIDECHAR 1

#include <curses.h>

#include <panel.h>

**namespace** display **{**

/\* Блок меню \*/

**typedef** struct menu\_t **{**

/\* Пункты \*/

const wchar\_t**\*\*** options**;**

/\* Длина \*/

int length**;**

/\* Обработчик \*/

bool **(\***key**)(**int**);**

**}** menu\_t**;**

/\* Блок данных \*/

**typedef** struct data\_t **{**

/\* Отрисовщик \*/

void **(\***draw**)();**

/\* Обработчик \*/

bool **(\***key**)(**int**);**

**}** data\_t**;**

/\* Сообщение \*/

**typedef** struct message\_t **{**

/\* Заголовок \*/

const wchar\_t**\*** title**;**

/\* Текст \*/

const wchar\_t**\*** text**;**

**}** message\_t**;**

/\* Переменные сообщений \*/

extern const message\_t message\_nofile**;**

extern const message\_t message\_cantwrite**;**

extern const message\_t message\_nomemory**;**

extern const message\_t message\_noelement**;**

extern const message\_t message\_nothingfound**;**

extern const message\_t message\_author**;**

void init\_display**();**

void close\_display**();**

void go\_home**();**

void draw**();**

void draw\_menu**();**

void draw\_data**();**

void toggle\_menu**();**

void switch\_menu**(**menu\_t**,**int**=**0**);**

void switch\_data**(**data\_t**);**

void show\_message**(**message\_t**);**

bool key**();**

bool key\_menu**(**int**);**

bool key\_menu\_index**(**int**);**

bool key\_menu\_list**(**int**);**

bool key\_menu\_find**(**int**);**

bool key\_menu\_sort**(**int**);**

bool key\_menu\_save**(**int**);**

bool key\_menu\_read**(**int**);**

bool key\_menu\_process**(**int**);**

bool key\_menu\_home**(**int**);**

bool key\_data**(**int**);**

bool key\_data\_list**(**int**);**

bool key\_data\_process**(**int**);**

bool key\_data\_find\_guild**(**int**);**

bool key\_data\_find\_name**(**int**);**

bool key\_data\_form**(**int**);**

void draw\_data\_list**();**

void draw\_data\_process**();**

void draw\_data\_find\_guild**();**

void draw\_data\_find\_name**();**

void draw\_data\_form**();**

void draw\_message**();**

**}**

#endif /\* DISPLAY\_H \*/

1. **ФАЙЛ *DISPLAY.CPP***

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "display.h"

#include "list.h"

#include "form.h"

**namespace** display **{**

/\* Размеры окна меню (высота, ширина) \*/

const int menu\_height **=** 25**;**

const int menu\_width **=** 17**;**

/\* Размеры окна данных (высота, ширина) \*/

const int data\_height **=** 25**;**

const int data\_width **=** 80**;**

/\* Размеры окна сообщения (высота, ширина) \*/

const int message\_height **=** 7**;**

const int message\_width **=** 50**;**

/\* Позиция курсора на экране \*/

int cursor\_x **=** 79**;**

int cursor\_y **=** 24**;**

/\* Состояние меню (показывается/скрыто) \*/

bool menu\_hidden **=** **false;**

/\* Состояние сообщения (показывается/скрыто) \*/

bool message\_hidden **=** **true;**

/\* Кол-во запросов на перерисовку окна данных \*/

int redraw\_data **=** 2**;**

/\* Выбранный элемент основного списка на экране \*/

int data\_list\_selected **=** 0**;**

/\* Отступ с которым отображается основной список на экране \*/

int data\_list\_offset **=** 0**;**

/\* Выбранный элемент списка результатов на экране \*/

int data\_process\_selected **=** 0**;**

/\* Отступ с которым отображается список результатов на экране \*/

int data\_process\_offset **=** 0**;**

/\* Поле для поиска по цеху \*/

wchar\_t data\_find\_guild\_field**[**16**];**

int data\_find\_guild\_index **=** 0**;**

/\* Поле для поиска по имени \*/

wchar\_t data\_find\_name\_field**[**16**];**

int data\_find\_name\_index **=** 0**;**

/\* Окна меню, данных и сообщений \*/

WINDOW**\*** menu\_win**;**

WINDOW**\*** data\_win**;**

WINDOW**\*** message\_win**;**

/\* Панели меню и сообщений \*/

PANEL**\*** manu\_pan**;**

PANEL**\*** message\_pan**;**

/\* Строковые значения пунктов меню \*/

const wchar\_t**\*** menu\_index\_options**[]** **=** **{**

L"Добавление"**,**

L"Сортировка"**,**

L"Поиск"**,**

L"Изменение"**,**

L"Удаление"**,**

L"Обработка"**,**

L"Очистка"**,**

L"Сохранение"**,**

L"Загрузка"**,**

L"Выход"**,**

**};**

const wchar\_t**\*** menu\_list\_options**[]** **=** **{**

L"← Назад"**,**

L"Изменение"**,**

L"Удаление"**,**

**};**

const wchar\_t**\*** menu\_find\_options**[]** **=** **{**

L"← Назад"**,**

L"По имени"**,**

L"По номеру цеха"**,**

**};**

const wchar\_t**\*** menu\_sort\_options**[]** **=** **{**

L"← Назад"**,**

L"По > ID"**,**

L"По < ID"**,**

L"По > имени"**,**

L"По < имени"**,**

L"По > профессии"**,**

L"По < профессии"**,**

L"По > зарплате"**,**

L"По < зарплате"**,**

**};**

const wchar\_t**\*** menu\_save\_options**[]** **=** **{**

L"← Назад"**,**

L"В текстовый"**,**

L"В бинарный"**,**

**};**

const wchar\_t**\*** menu\_read\_options**[]** **=** **{**

L"← Назад"**,**

L"Из текстового"**,**

L"Из бинарного"**,**

**};**

const wchar\_t**\*** menu\_process\_options**[]** **=** **{**

L"← Назад"**,**

L"Сохранить"**,**

**};**

const wchar\_t**\*** menu\_home\_options**[]** **=** **{**

L"← Назад"**,**

**};**

/\* Содержимое меню: главное окно \*/

const menu\_t menu\_index **=** **{**

**.**options **=** menu\_index\_options**,**

**.**length **=** 10**,**

**.**key **=** key\_menu\_index

**};**

/\* Содержимое меню: частного списка \*/

const menu\_t menu\_list **=** **{**

**.**options **=** menu\_list\_options**,**

**.**length **=** 3**,**

**.**key **=** key\_menu\_list

**};**

/\* Содержимое меню: поиска \*/

const menu\_t menu\_find **=** **{**

**.**options **=** menu\_find\_options**,**

**.**length **=** 3**,**

**.**key **=** key\_menu\_find

**};**

/\* Содержимое меню: сортировки \*/

const menu\_t menu\_sort **=** **{**

**.**options **=** menu\_sort\_options**,**

**.**length **=** 9**,**

**.**key **=** key\_menu\_sort

**};**

/\* Содержимое меню: сохранения \*/

const menu\_t menu\_save **=** **{**

**.**options **=** menu\_save\_options**,**

**.**length **=** 3**,**

**.**key **=** key\_menu\_save

**};**

/\* Содержимое меню: загрузки \*/

const menu\_t menu\_read **=** **{**

**.**options **=** menu\_read\_options**,**

**.**length **=** 3**,**

**.**key **=** key\_menu\_read

**};**

/\* Содержимое меню: обработка \*/

const menu\_t menu\_process **=** **{**

**.**options **=** menu\_process\_options**,**

**.**length **=** 2**,**

**.**key **=** key\_menu\_process

**};**

/\* Содержимое меню: возврата \*/

const menu\_t menu\_home **=** **{**

**.**options **=** menu\_home\_options**,**

**.**length **=** 1**,**

**.**key **=** key\_menu\_home

**};**

/\* Текущие содержимое меню \*/

menu\_t menu\_current **=** menu\_index**;**

/\* Текущий выбранный пункт в меню \*/

int menu\_selected **=** 0**;**

/\* Содержимое окна данных: список элементов \*/

const data\_t data\_list **=** **{**

**.**draw **=** draw\_data\_list**,**

**.**key **=** key\_data\_list

**};**

/\* Содержимое окна данных: результирующая таблица \*/

const data\_t data\_process **=** **{**

**.**draw **=** draw\_data\_process**,**

**.**key **=** key\_data\_process

**};**

/\* Содержимое окна данных: форма поиска по цеху \*/

const data\_t data\_find\_guild **=** **{**

**.**draw **=** draw\_data\_find\_guild**,**

**.**key **=** key\_data\_find\_guild

**};**

/\* Содержимое окна данных: форма поиска по имени \*/

const data\_t data\_find\_name **=** **{**

**.**draw **=** draw\_data\_find\_name**,**

**.**key **=** key\_data\_find\_name

**};**

/\* Содержимое окна данных: форма ввода элемента \*/

const data\_t data\_form **=** **{**

**.**draw **=** draw\_data\_form**,**

**.**key **=** key\_data\_form

**};**

/\* Текущие содержимое окна данных \*/

data\_t data\_current **=** data\_list**;**

/\* Содержимое сообщения: файл не найден \*/

const message\_t message\_nofile **=** **{**

**.**title **=** L"Ошибка"**,**

**.**text **=** L"Файл не найден"

**};**

/\* Содержимое сообщения: запись не возможна \*/

const message\_t message\_cantwrite **=** **{**

**.**title **=** L"Ошибка"**,**

**.**text **=** L"Данные не были сохранены"

**};**

/\* Содержимое сообщения: недостаточно памяти \*/

const message\_t message\_nomemory **=** **{**

**.**title **=** L"Ошибка"**,**

**.**text **=** L"Недостаточно памяти"

**};**

/\* Содержимое сообщения: искомый элемент не найтен \*/

const message\_t message\_noelement **=** **{**

**.**title **=** L"Ошибка"**,**

**.**text **=** L"Элемент не найден"

**};**

/\* Содержимое сообщения: не найдено ни одного элемента по этому запросу \*/

const message\_t message\_nothingfound **=** **{**

**.**title **=** L"Ошибка"**,**

**.**text **=** L"Ничего не найдено"

**};**

/\* Содержимое сообщения: автор программы \*/

const message\_t message\_author **=** **{**

**.**title **=** L"Курсовой проект по ОПиАЯ"**,**

**.**text **=** L"Автор: Куркчи Ариф ИСб-21"

**};**

/\* Текущие содержимое сообщения \*/

message\_t message\_current**;**

/\* Текущая форма ввода \*/

form**::**form\_t**\*** form\_current**;**

/\* Текущий результирующий список \*/

list**::**result\_t**\*** data\_process\_data **=** **NULL;**

/\* Текущий размер результирующего списка \*/

int data\_process\_length **=** 0**;**

/\* Настройка основных параметров экрана \*/

void init\_display**()** **{**

initscr**();**

clear**();**

noecho**();**

cbreak**();**

keypad**(**stdscr**,true);**

start\_color**();**

init\_pair**(**1**,**COLOR\_GREEN**,**COLOR\_BLACK**);**

init\_pair**(**2**,**COLOR\_BLACK**,**COLOR\_WHITE**);**

/\* Создание окон \*/

menu\_win **=** newwin**(**menu\_height**,** menu\_width**,** 0**,** 0**);**

data\_win **=** newwin**(**data\_height**,** data\_width**,** 0**,** 0**);**

message\_win **=** newwin**(**message\_height**,**message\_width**,(**25**-**message\_height**)/**2**-**1**,(**80**-**message\_width**)/**2**-**1**);**

/\* Создание панелей \*/

manu\_pan **=** new\_panel**(**menu\_win**);**

message\_pan **=** new\_panel**(**message\_win**);**

/\* Скрыть изначально панель сообщений \*/

hide\_panel**(**message\_pan**);**

wcscpy**(**data\_find\_guild\_field**,**L""**);**

wcscpy**(**data\_find\_name\_field**,** L""**);**

form\_current **=** form**::**full\_form**;**

**}**

/\* Очистка и выключение экрана \*/

void close\_display**()** **{**

clear**();**

refresh**();**

endwin**();**

**}**

/\* Возврат к главному экрану \*/

void go\_home**()** **{**

switch\_menu**(**menu\_index**,**0**);**

switch\_data**(**data\_list**);**

**}**

/\* Отрисовка экрана \*/

void draw**()** **{**

**if(!**message\_hidden**)** **{**

draw\_message**();**

**}** **else** **{**

/\* Проверка необходимости перерисовки данных \*/

**if(**menu\_hidden **||** redraw\_data **!=** 0**)** **{**

**if(**redraw\_data **<** 0**)** **{**

wclear**(**data\_win**);**

redraw\_data**++;**

**}** **else** **{**

redraw\_data**--;**

**}**

draw\_data**();**

**}**

**if(!**menu\_hidden**)** **{**

draw\_menu**();**

**}**

**}**

/\* Передвижение курсора в нужное положение \*/

move**(**cursor\_y**,**cursor\_x**);**

refresh**();**

**}**

/\* Отрисовка меню \*/

void draw\_menu**()** **{**

wbkgd**(**menu\_win**,**COLOR\_PAIR**(**1**));**

/\* Отрисовка верхней линии \*/

wattron**(**menu\_win**,**COLOR\_PAIR**(**1**));**

**for(**int i**=**0**;**i**<=**menu\_width**-**2**;**i**++)** **{**

mvwadd\_wch**(**menu\_win**,**0**,**i**,**WACS\_HLINE**);**

**}**

/\* Отрисовка боковой линии \*/

mvwadd\_wch**(**menu\_win**,**0**,**menu\_width**-**1**,**WACS\_URCORNER**);**

**for(**int i**=**1**;**i**<=**menu\_height**-**2**;**i**++)** **{**

mvwadd\_wch**(**menu\_win**,**i**,**menu\_width**-**1**,**WACS\_VLINE**);**

**}**

/\* Отрисовка нижней линии \*/

mvwadd\_wch**(**menu\_win**,**menu\_height**-**1**,**menu\_width**-**1**,**WACS\_LRCORNER**);**

**for(**int i**=**0**;**i**<=**menu\_width**-**2**;**i**++)** **{**

mvwadd\_wch**(**menu\_win**,**menu\_height**-**1**,**i**,**WACS\_HLINE**);**

**}**

int y **=** 1**;**

wchar\_t tmp**[**17**];**

/\* Отрисовка пунктов меню \*/

**for(**int i**=**0**;**i**<**menu\_current**.**length**;++**i**)** **{**

swprintf**(**tmp**,**17**,**L" %-14S "**,**menu\_current**.**options**[**i**]);**

**if(**menu\_selected **==** i**)** **{**

wattron**(**menu\_win**,**A\_REVERSE**);**

mvwprintw**(**menu\_win**,**y**,**0**,**"%S"**,**tmp**);**

wattroff**(**menu\_win**,**A\_REVERSE**);**

**}** **else** **{**

mvwprintw**(**menu\_win**,**y**,**0**,**"%S"**,**tmp**);**

**}**

y **+=** 2**;**

**}**

wattroff**(**menu\_win**,**COLOR\_PAIR**(**1**));**

/\* Отрисовка вспомогательной информации \*/

mvwprintw**(**menu\_win**,**22**,**2**,**"%S"**,**L"Вкл/Выкл меню"**);**

wattron**(**menu\_win**,**COLOR\_PAIR**(**2**));**

mvwprintw**(**menu\_win**,**23**,**6**,**"%S"**,**L"TAB"**);**

wattroff**(**menu\_win**,**COLOR\_PAIR**(**2**));**

wrefresh**(**menu\_win**);**

**}**

/\* Отрисовка данных \*/

void draw\_data**()** **{**

/\* Делегация отрисовки меню \*/

data\_current**.**draw**();**

/\* Отрисовка указателя на раскрытие меню \*/

**if(**menu\_hidden**)** **{**

wattron**(**data\_win**,**COLOR\_PAIR**(**2**));**

mvwprintw**(**data\_win**,**1**,**0**,**"%S"**,**L"→"**);**

mvwprintw**(**data\_win**,**2**,**0**,**"%S"**,**L"T"**);**

mvwprintw**(**data\_win**,**3**,**0**,**"%S"**,**L"A"**);**

mvwprintw**(**data\_win**,**4**,**0**,**"%S"**,**L"B"**);**

wattroff**(**data\_win**,**COLOR\_PAIR**(**2**));**

**}**

wrefresh**(**data\_win**);**

**}**

/\* Показать/скрыть меню \*/

void toggle\_menu**()** **{**

**if(**menu\_hidden**)** **{**

show\_panel**(**manu\_pan**);**

**}** **else** **{**

hide\_panel**(**manu\_pan**);**

**}**

menu\_hidden **=** **!**menu\_hidden**;**

update\_panels**();**

doupdate**();**

**}**

/\* Переключение отображаемое меню \*/

void switch\_menu**(**menu\_t menu**,**int selection**)** **{**

wclear**(**menu\_win**);**

menu\_selected **=** selection**;**

menu\_current **=** menu**;**

redraw\_data **=** 1**;**

**}**

/\* Переключение отображаемых данных \*/

void switch\_data**(**data\_t data**)** **{**

wclear**(**data\_win**);**

data\_current **=** data**;**

redraw\_data **=** 1**;**

**}**

/\* Отображение сообщения \*/

void show\_message**(**message\_t m**)** **{**

message\_current **=** m**;**

message\_hidden **=** **false;**

show\_panel**(**message\_pan**);**

**}**

/\* Обработка нажатий клавиш \*/

bool key**()** **{**

cursor\_x **=** 79**;**

cursor\_y **=** 24**;**

wint\_t wc**;**

wget\_wch**(**stdscr**,&**wc**);**

/\* Проверка нажатия любой клавиши в сообщении \*/

**if(!**message\_hidden**)** **{**

message\_hidden **=** **true;**

hide\_panel**(**message\_pan**);**

draw\_data**();**

draw\_menu**();**

refresh**();**

**return** **true;**

**}**

**switch(**wc**)** **{**

**case** KEY\_HELP**:**

**case** KEY\_F**(**1**):** // Помощь

show\_message**(**message\_author**);**

redraw\_data**++;**

**break;**

**case** L'\t'**:** // Показать/скрыть меню

toggle\_menu**();**

**break;**

**default:** // Делегация обработки клавиши обработчикам

**if(!**menu\_hidden**)** **{**

**return** key\_menu**(**wc**);**

**}** **else** **{**

**return** key\_data**(**wc**);**

**}**

**break;**

**}**

**return** **true;**

**}**

/\* Обработка клавиши в меню \*/

bool key\_menu**(**int c**)** **{**

**switch(**c**)** **{**

**case** KEY\_UP**:** // Вверх

**if(**menu\_selected **==** 0**)** **{**

menu\_selected **=** menu\_current**.**length**-**1**;**

**}** **else** **{**

menu\_selected**--;**

**}**

**break;**

**case** KEY\_DOWN**:** // Вниз

**if(**menu\_selected **==** menu\_current**.**length**-**1**)** **{**

menu\_selected **=** 0**;**

**}** **else** **{**

menu\_selected**++;**

**}**

**break;**

**case** KEY\_ENTER**:** // Делегация обработки пункта

**case** '\n'**:**

**return** menu\_current**.**key**(**menu\_selected**);**

**break;**

**}**

**return** **true;**

**}**

/\* Обработка пункта в главном меню \*/

bool key\_menu\_index**(**int c**)** **{**

list**::**node\_t**\*** tmp**;**

**switch(**c**)** **{**

**case** 0**:** // Добавление

switch\_menu**(**menu\_home**,**0**);**

switch\_data**(**data\_form**);**

form**::**clear**(**form\_current**);**

form\_current**->**submit **=** form**::**add\_submit**;**

toggle\_menu**();**

**break;**

**case** 1**:** // Соритровки

switch\_menu**(**menu\_sort**,**1**);**

**break;**

**case** 2**:** // Поиск

switch\_menu**(**menu\_find**,**1**);**

**break;**

**case** 3**:** // Изменение

**if(!**list**::**global**.**length**)** **{**

show\_message**(**message\_noelement**);**

**break;**

**}**

switch\_menu**(**menu\_home**,**0**);**

switch\_data**(**data\_form**);**

/\* Заполнение формы данными \*/

form**::**clear**(**form\_current**);**

tmp **=** list**::**get**(\***list**::**current**,**data\_list\_selected**);**

form\_current**->**field**[**0**]->**data **=** list**::**itow**(**tmp**->**data**.**id**,**form\_current**->**field**[**0**]->**length**);**

wcscpy**(**form\_current**->**field**[**1**]->**data**,**tmp**->**data**.**name**);**

form\_current**->**field**[**2**]->**data **=** list**::**itow**(**tmp**->**data**.**b\_year**,**form\_current**->**field**[**2**]->**length**);**

swprintf**(**form\_current**->**field**[**3**]->**data**,**form\_current**->**field**[**3**]->**length**+**1**,**L"%C"**,**tmp**->**data**.**sex**);**

wcscpy**(**form\_current**->**field**[**4**]->**data**,**tmp**->**data**.**prof**);**

form\_current**->**field**[**5**]->**data **=** list**::**itow**(**tmp**->**data**.**exp**,**form\_current**->**field**[**5**]->**length**);**

form\_current**->**field**[**6**]->**data **=** list**::**itow**(**tmp**->**data**.**rank**,**form\_current**->**field**[**6**]->**length**);**

form\_current**->**field**[**7**]->**data **=** list**::**itow**(**tmp**->**data**.**guild**,**form\_current**->**field**[**7**]->**length**);**

form\_current**->**field**[**8**]->**data **=** list**::**itow**(**tmp**->**data**.**sect**,**form\_current**->**field**[**8**]->**length**);**

form\_current**->**field**[**9**]->**data **=** list**::**ftow**(**L"%.2f"**,**tmp**->**data**.**salary**,**form\_current**->**field**[**9**]->**length**);**

form\_current**->**field**[**0**]->**index **=** wcslen**(**form\_current**->**field**[**0**]->**data**);**

form\_current**->**field**[**1**]->**index **=** wcslen**(**form\_current**->**field**[**1**]->**data**);**

form\_current**->**field**[**2**]->**index **=** wcslen**(**form\_current**->**field**[**2**]->**data**);**

form\_current**->**field**[**3**]->**index **=** wcslen**(**form\_current**->**field**[**3**]->**data**);**

form\_current**->**field**[**4**]->**index **=** wcslen**(**form\_current**->**field**[**4**]->**data**);**

form\_current**->**field**[**5**]->**index **=** wcslen**(**form\_current**->**field**[**5**]->**data**);**

form\_current**->**field**[**6**]->**index **=** wcslen**(**form\_current**->**field**[**6**]->**data**);**

form\_current**->**field**[**7**]->**index **=** wcslen**(**form\_current**->**field**[**7**]->**data**);**

form\_current**->**field**[**8**]->**index **=** wcslen**(**form\_current**->**field**[**8**]->**data**);**

form\_current**->**field**[**9**]->**index **=** wcslen**(**form\_current**->**field**[**9**]->**data**);**

form\_current**->**submit **=** form**::**edit\_submit**;**

form**::**current\_el **=** tmp**->**index**;**

toggle\_menu**();**

**break;**

**case** 4**:** // Удаление

**if(!**list**::**global**.**length**)** **{**

show\_message**(**message\_noelement**);**

**break;**

**}**

tmp **=** list**::**get**(**list**::**global**,**data\_list\_selected**);**

**if(!**list**::**erase**(**list**::**global**,**tmp**->**index**))** **{**

show\_message**(**message\_noelement**);**

**}**

redraw\_data **=** **-**1**;**

**break;**

**case** 5**:** // Обработка

**if(**data\_process\_data **!=** **NULL)** **{**

**delete** **[]** data\_process\_data**;**

**}**

data\_process\_data **=** list**::**process**(**list**::**global**,**data\_process\_length**);**

data\_process\_offset **=** 0**;**

data\_process\_selected **=** 0**;**

switch\_menu**(**menu\_process**,**1**);**

switch\_data**(**data\_process**);**

toggle\_menu**();**

**break;**

**case** 6**:** // Очистка

list**::**free**(**list**::**global**);**

redraw\_data **=** **-**1**;**

**break;**

**case** 7**:** // Сохранение

switch\_menu**(**menu\_save**,**1**);**

**break;**

**case** 8**:** // Загрузка

switch\_menu**(**menu\_read**,**1**);**

**break;**

**case** 9**:** // Выход

**return** **false;**

**break;**

**}**

**return** **true;**

**}**

/\* Обработка пункта в отображении списка \*/

bool key\_menu\_list**(**int c**)** **{**

list**::**node\_t**\*** tmp**;**

**switch(**c**)** **{**

**case** 1**:** // Изменить

**if(!**list**::**current**->**length**)** **{**

show\_message**(**message\_noelement**);**

**break;**

**}**

switch\_menu**(**menu\_home**,**0**);**

switch\_data**(**data\_form**);**

/\* Заполнение формы данными \*/

form**::**clear**(**form\_current**);**

tmp **=** list**::**get**(\***list**::**current**,**data\_list\_selected**);**

form\_current**->**field**[**0**]->**data **=** list**::**itow**(**tmp**->**data**.**id**,**form\_current**->**field**[**0**]->**length**);**

wcscpy**(**form\_current**->**field**[**1**]->**data**,**tmp**->**data**.**name**);**

form\_current**->**field**[**2**]->**data **=** list**::**itow**(**tmp**->**data**.**b\_year**,**form\_current**->**field**[**2**]->**length**);**

swprintf**(**form\_current**->**field**[**3**]->**data**,**form\_current**->**field**[**3**]->**length**+**1**,**L"%C"**,**tmp**->**data**.**sex**);**

wcscpy**(**form\_current**->**field**[**4**]->**data**,**tmp**->**data**.**prof**);**

form\_current**->**field**[**5**]->**data **=** list**::**itow**(**tmp**->**data**.**exp**,**form\_current**->**field**[**5**]->**length**);**

form\_current**->**field**[**6**]->**data **=** list**::**itow**(**tmp**->**data**.**rank**,**form\_current**->**field**[**6**]->**length**);**

form\_current**->**field**[**7**]->**data **=** list**::**itow**(**tmp**->**data**.**guild**,**form\_current**->**field**[**7**]->**length**);**

form\_current**->**field**[**8**]->**data **=** list**::**itow**(**tmp**->**data**.**sect**,**form\_current**->**field**[**8**]->**length**);**

form\_current**->**field**[**9**]->**data **=** list**::**ftow**(**L"%.2f"**,**tmp**->**data**.**salary**,**form\_current**->**field**[**9**]->**length**);**

form\_current**->**field**[**0**]->**index **=** wcslen**(**form\_current**->**field**[**0**]->**data**);**

form\_current**->**field**[**1**]->**index **=** wcslen**(**form\_current**->**field**[**1**]->**data**);**

form\_current**->**field**[**2**]->**index **=** wcslen**(**form\_current**->**field**[**2**]->**data**);**

form\_current**->**field**[**3**]->**index **=** wcslen**(**form\_current**->**field**[**3**]->**data**);**

form\_current**->**field**[**4**]->**index **=** wcslen**(**form\_current**->**field**[**4**]->**data**);**

form\_current**->**field**[**5**]->**index **=** wcslen**(**form\_current**->**field**[**5**]->**data**);**

form\_current**->**field**[**6**]->**index **=** wcslen**(**form\_current**->**field**[**6**]->**data**);**

form\_current**->**field**[**7**]->**index **=** wcslen**(**form\_current**->**field**[**7**]->**data**);**

form\_current**->**field**[**8**]->**index **=** wcslen**(**form\_current**->**field**[**8**]->**data**);**

form\_current**->**field**[**9**]->**index **=** wcslen**(**form\_current**->**field**[**9**]->**data**);**

form\_current**->**submit **=** form**::**edit\_submit**;**

form**::**current\_el **=** tmp**->**index**;**

toggle\_menu**();**

**break;**

**case** 2**:** // Удалить

**if(!**list**::**current**->**length**)** **{**

show\_message**(**message\_noelement**);**

**break;**

**}**

list**::**node\_t**\*** tmp **=** list**::**get**(\***list**::**current**,**data\_list\_selected**);**

**if(!**list**::**erase**(**list**::**global**,**tmp**->**index**))** **{**

show\_message**(**message\_noelement**);**

**}**

redraw\_data **=** **-**1**;**

**break;**

**}**

list**::**free**(\***list**::**current**);**

list**::**current **=** **&**list**::**global**;**

switch\_menu**(**menu\_index**,**0**);**

**return** **true;**

**}**

/\* Обработка пункта в меню поиска \*/

bool key\_menu\_find**(**int c**)** **{**

**switch(**c**)** **{**

**case** 1**:** // По названию

switch\_data**(**data\_find\_name**);**

switch\_menu**(**menu\_home**,**0**);**

toggle\_menu**();**

**break;**

**case** 2**:** // По номеру цеха

switch\_data**(**data\_find\_guild**);**

switch\_menu**(**menu\_home**,**0**);**

toggle\_menu**();**

**break;**

**default:**

switch\_menu**(**menu\_index**,**2**);**

**break;**

**}**

**return** **true;**

**}**

/\* Обработка пункта в меню сотритовки \*/

bool key\_menu\_sort**(**int c**)** **{**

**switch(**c**)** **{**

**case** 1**:** // По > ID

list**::**sort**(**list**::**global**,**list**::**compare\_by\_id**,false);**

**break;**

**case** 2**:** // По < ID

list**::**sort**(**list**::**global**,**list**::**compare\_by\_id**,true);**

**break;**

**case** 3**:** // По > имени

list**::**sort**(**list**::**global**,**list**::**compare\_by\_name**,false);**

**break;**

**case** 4**:** // По < имени

list**::**sort**(**list**::**global**,**list**::**compare\_by\_name**,true);**

**break;**

**case** 5**:** // По > профессии

list**::**sort**(**list**::**global**,**list**::**compare\_by\_prof**,false);**

**break;**

**case** 6**:** // По < профессии

list**::**sort**(**list**::**global**,**list**::**compare\_by\_prof**,true);**

**break;**

**case** 7**:** // По > зарплате

list**::**sort**(**list**::**global**,**list**::**compare\_by\_salary**,false);**

**break;**

**case** 8**:** // По < зарплате

list**::**sort**(**list**::**global**,**list**::**compare\_by\_salary**,true);**

**break;**

**}**

data\_list\_selected **=** 0**;**

data\_list\_offset **=** 0**;**

switch\_menu**(**menu\_index**,**1**);**

**return** **true;**

**}**

/\* Обработка пункта в меню сохранения \*/

bool key\_menu\_save**(**int c**)** **{**

**switch(**c**)** **{**

**case** 1**:** // В текстовый

**if(!**list**::**save**(**list**::**global**,**"data.txt"**,false))** **{**

show\_message**(**message\_cantwrite**);**

**}**

**break;**

**case** 2**:** // В бинарный

**if(!**list**::**save**(**list**::**global**,**"data.bin"**,true))** **{**

show\_message**(**message\_cantwrite**);**

**}**

**break;**

**}**

switch\_menu**(**menu\_index**,**0**);**

**return** **true;**

**}**

/\* Обработка пункта в меню загрузки \*/

bool key\_menu\_read**(**int c**)** **{**

**switch(**c**)** **{**

**case** 1**:** // Из текстового

list**::**free**(**list**::**global**);**

**if(!**list**::**read**(**list**::**global**,**"data.txt"**,false))** **{**

show\_message**(**message\_nofile**);**

**}**

data\_list\_selected **=** 0**;**

data\_list\_offset **=** 0**;**

**break;**

**case** 2**:** // Из бинарного

list**::**free**(**list**::**global**);**

**if(!**list**::**read**(**list**::**global**,**"data.bin"**,true))** **{**

show\_message**(**message\_nofile**);**

**}**

data\_list\_selected **=** 0**;**

data\_list\_offset **=** 0**;**

**break;**

**}**

switch\_menu**(**menu\_index**,**0**);**

**return** **true;**

**}**

/\* Обработка пункта в меню выходной таблицы \*/

bool key\_menu\_process**(**int c**)** **{**

**switch(**c**)** **{**

**case** 0**:** // Назад

go\_home**();**

**break;**

**case** 1**:** // Сохранить

**if(!**list**::**save**(**data\_process\_data**,**data\_process\_length**,**"result.txt"**))** **{**

show\_message**(**message\_cantwrite**);**

**}**

**break;**

**}**

**return** **true;**

**}**

/\* Обработка пункта в меню возврата на главную \*/

bool key\_menu\_home**(**int c**)** **{**

switch\_menu**(**menu\_index**,**0**);**

switch\_data**(**data\_list**);**

data\_list\_selected **=** 0**;**

data\_list\_offset **=** 0**;**

**return** **true;**

**}**

/\* Обработка клавиши в окне данных \*/

bool key\_data**(**int c**)** **{**

**switch(**c**)** **{**

**default:**

/\* Делегация обработки клавиши \*/

**return** data\_current**.**key**(**c**);**

**break;**

**}**

**return** **true;**

**}**

/\* Обработка клавиши в данных списка \*/

bool key\_data\_list**(**int c**)** **{**

**switch(**c**)** **{**

**case** KEY\_UP**:** // Вверх

**if(**data\_list\_selected **!=** 0**)** **{**

data\_list\_selected**--;**

**}**

**break;**

**case** KEY\_DOWN**:** // Вниз

**if(**data\_list\_selected **<** list**::**current**->**length**-**1**)** **{**

data\_list\_selected**++;**

**}**

**break;**

**default:**

**break;**

**}**

**if(**data\_list\_selected **<** data\_list\_offset**)** **{**

data\_list\_offset**--;**

**}**

**if(**data\_list\_selected **>** data\_list\_offset**+**10**)** **{**

data\_list\_offset**++;**

**}**

**return** **true;**

**}**

/\* Обработка клавиши в данных выходной таблицы \*/

bool key\_data\_process**(**int c**)** **{**

**switch(**c**)** **{**

**case** KEY\_UP**:** // Вверх

**if(**data\_process\_selected **!=** 0**)** **{**

data\_process\_selected**--;**

**}**

**break;**

**case** KEY\_DOWN**:** // Вниз

**if(**data\_process\_selected **<** data\_process\_length**-**1**)** **{**

data\_process\_selected**++;**

**}**

**break;**

**default:**

**break;**

**}**

**if(**data\_process\_selected **<** data\_process\_offset**)** **{**

data\_process\_offset**--;**

**}**

**if(**data\_process\_selected **>** data\_process\_offset**+**9**)** **{**

data\_process\_offset**++;**

**}**

**return** **true;**

**}**

/\* Обработка клавиши в данных поиска по цеху \*/

bool key\_data\_find\_guild**(**int c**)** **{**

list**::**data\_t to\_find**;**

to\_find**.**id **=** **-**1**;**

wcscpy**(**to\_find**.**name**,**L""**);**

to\_find**.**b\_year **=** **-**1**;**

to\_find**.**sex **=** L'\0'**;**

wcscpy**(**to\_find**.**prof**,**L""**);**

to\_find**.**exp **=** **-**1**;**

to\_find**.**rank **=** **-**1**;**

to\_find**.**guild **=** **-**1**;**

to\_find**.**sect **=** **-**1**;**

to\_find**.**salary **=** **-**1**;**

**switch(**c**)** **{**

**case** KEY\_BACKSPACE**:** // Стереть

**case** 127**:**

**if(**data\_find\_guild\_index **!=** 0**)** **{**

data\_find\_guild\_field**[--**data\_find\_guild\_index**]** **=** L'\0'**;**

**}**

**break;**

**case** KEY\_ENTER**:** // Найти

**case** L'\n'**:**

to\_find**.**guild **=** list**::**wtoi**(**data\_find\_guild\_field**);**

list**::**current **=** list**::**find**(**list**::**global**,**to\_find**);**

**if(!**list**::**current**->**length**)** **{**

show\_message**(**message\_nothingfound**);**

list**::**current **=** **&**list**::**global**;**

switch\_menu**(**menu\_index**,**0**);**

**}** **else** **{**

switch\_menu**(**menu\_list**,**1**);**

**}**

switch\_data**(**data\_list**);**

**if(**menu\_hidden**)** **{**

toggle\_menu**();**

**}**

**break;**

**default:** // Ввод символа

**if(**data\_find\_guild\_index **<** 9 **&&** **(**c **>=** L'0' **&&** c **<=** L'9'**))** **{**

data\_find\_guild\_field**[**data\_find\_guild\_index**++]** **=** c**;**

data\_find\_guild\_field**[**data\_find\_guild\_index**]** **=** L'\0'**;**

**}**

**break;**

**}**

**return** **true;**

**}**

/\* Обработка клавиши в данных поиска по имени \*/

bool key\_data\_find\_name**(**int c**)** **{**

list**::**data\_t to\_find**;**

to\_find**.**id **=** **-**1**;**

wcscpy**(**to\_find**.**name**,**L""**);**

to\_find**.**b\_year **=** **-**1**;**

to\_find**.**sex **=** L'\0'**;**

wcscpy**(**to\_find**.**prof**,**L""**);**

to\_find**.**exp **=** **-**1**;**

to\_find**.**rank **=** **-**1**;**

to\_find**.**guild **=** **-**1**;**

to\_find**.**sect **=** **-**1**;**

to\_find**.**salary **=** **-**1**;**

**switch(**c**)** **{**

**case** KEY\_BACKSPACE**:** // Стереть

**case** 127**:**

**if(**data\_find\_name\_index **!=** 0**)** **{**

data\_find\_name\_field**[--**data\_find\_name\_index**]** **=** L'\0'**;**

**}**

**break;**

**case** KEY\_ENTER**:** // Найти

**case** L'\n'**:**

wcscpy**(**to\_find**.**name**,**data\_find\_name\_field**);**

list**::**current **=** list**::**find**(**list**::**global**,**to\_find**);**

**if(!**list**::**current**->**length**)** **{**

show\_message**(**message\_nothingfound**);**

list**::**current **=** **&**list**::**global**;**

switch\_menu**(**menu\_index**,**0**);**

**}** **else** **{**

switch\_menu**(**menu\_list**,**1**);**

**}**

switch\_data**(**data\_list**);**

**if(**menu\_hidden**)** **{**

toggle\_menu**();**

**}**

**break;**

**default:** // Ввод символа

**if(**data\_find\_name\_index **<** 17 **&&** **(**

**(**c **>=** L'0' **&&** c **<=** L'9'**)** **||**

**(**c **>=** L'a' **&&** c **<=** L'z'**)** **||**

**(**c **>=** L'A' **&&** c **<=** L'Z'**)** **||**

**(**c **>=** L'а' **&&** c **<=** L'я'**)** **||**

**(**c **>=** L'А' **&&** c **<=** L'Я'**)** **||**

**(**c **==** L'ё' **||** c **==** L'Ё'**)** **||**

**(**c **==** L' ' **||** c **==** L'-' **||** c **==** L'\_'**)**

**))** **{**

data\_find\_name\_field**[**data\_find\_name\_index**++]** **=** c**;**

data\_find\_name\_field**[**data\_find\_name\_index**]** **=** L'\0'**;**

**}**

**break;**

**}**

**return** **true;**

**}**

/\* Обработка клавиши в данных формы \*/

bool key\_data\_form**(**int c**)** **{**

/\* Делегация клавиши в форму \*/

form**::**key**(**form\_current**,**c**);**

**return** **true;**

**}**

/\* Отрисовка списка данных \*/

void draw\_data\_list**()** **{**

/\* Заголовок \*/

mvwprintw**(**data\_win**,**0**,**0**,**"%S"**,**L"┌───┬────────────────────┬────┬───┬──────────┬────┬──────┬───┬───────┬─────────┐"**);**

mvwprintw**(**data\_win**,**1**,**0**,**"%S"**,**L"│ ID│ Имя│ Год│Пол│ Профессия│Опыт│Разряд│Цех│Участок│ Зарплата│"**);**

int y **=** 2**;**

/\* Отрисовка строки \*/

list**::**node\_t**\*** ptr **=** list**::**get**(\***list**::**current**,**data\_list\_offset**);**

**while(**ptr **!=** **NULL** **&&** y **<** 25**)** **{**

mvwprintw**(**data\_win**,**y**,**0**,**"%S"**,**L"├───┼────────────────────┼────┼───┼──────────┼────┼──────┼───┼───────┼─────────┤"**);**

wchar\_t tmp**[**81**];**

swprintf**(**tmp**,**81**,**L"│%3d│%20S│%4d│%3C│%10S│%4d│%6d│%3d│%7d│%9.2f│\n"**,**

ptr**->**data**.**id**,**

ptr**->**data**.**name**,**

ptr**->**data**.**b\_year**,**

ptr**->**data**.**sex**,**

ptr**->**data**.**prof**,**

ptr**->**data**.**exp**,**

ptr**->**data**.**rank**,**

ptr**->**data**.**guild**,**

ptr**->**data**.**sect**,**

ptr**->**data**.**salary**);**

y**++;**

**if(**data\_list\_offset **+** **(**y**-**1**)/**2 **-** 1 **==** data\_list\_selected**)** **{**

wattron**(**data\_win**,**COLOR\_PAIR**(**2**));**

mvwprintw**(**data\_win**,**y**,**0**,**"%S"**,**tmp**);**

wattroff**(**data\_win**,**COLOR\_PAIR**(**2**));**

**}** **else** **{**

mvwprintw**(**data\_win**,**y**,**0**,**"%S"**,**tmp**);**

**}**

y**++;**

ptr **=** ptr**->**next**;**

**}**

mvwprintw**(**data\_win**,**y**,**0**,**"%S"**,**L"└───┴────────────────────┴────┴───┴──────────┴────┴──────┴───┴───────┴─────────┘"**);**

**}**

/\* Отрисовка выходной таблицы \*/

void draw\_data\_process**()** **{**

wchar\_t line**[**81**];**

/\* Заголовок \*/

mvwprintw**(**data\_win**,**0**,**0**,**"%S"**,**L"┌───┬─────────────────────────────────────────────────────────────────┬───────┐"**);**

mvwprintw**(**data\_win**,**1**,**0**,**"%S"**,**L"│ │ Количество рабочих по стажу лет │ │"**);**

mvwprintw**(**data\_win**,**2**,**0**,**"%S"**,**L"│Цех├──────────┬──────────┬──────────┬──────────┬──────────┬──────────┤ Всего │"**);**

mvwprintw**(**data\_win**,**3**,**0**,**"%S"**,**L"│ │ До 6 │ 6-10 │ 11-15 │ 16-20 │ 21-25 │ Свыше 26 │ │"**);**

int y **=** 4**;**

/\* Отрисовка строки \*/

int i **=** data\_process\_offset**;**

**while(**i **<** data\_process\_length **&&** y **<** 25**)** **{**

mvwprintw**(**data\_win**,**y**,**0**,**"%S"**,**L"├───┼──────────┼──────────┼──────────┼──────────┼──────────┼──────────┼───────┤"**);**

y**++;**

swprintf**(**line**,**81**,**L"│%3d│%10d│%10d│%10d│%10d│%10d│%10d│%7d│"**,**

data\_process\_data**[**i**].**guild**,**

data\_process\_data**[**i**].**cat1**,**

data\_process\_data**[**i**].**cat2**,**

data\_process\_data**[**i**].**cat3**,**

data\_process\_data**[**i**].**cat4**,**

data\_process\_data**[**i**].**cat5**,**

data\_process\_data**[**i**].**cat6**,**

data\_process\_data**[**i**].**total**);**

**if(**data\_process\_offset **+** **(**y**-**3**)/**2 **-** 1 **==** data\_process\_selected**)** **{**

wattron**(**data\_win**,**COLOR\_PAIR**(**2**));**

mvwprintw**(**data\_win**,**y**,**0**,**"%S"**,**line**);**

wattroff**(**data\_win**,**COLOR\_PAIR**(**2**));**

**}** **else** **{**

mvwprintw**(**data\_win**,**y**,**0**,**"%S"**,**line**);**

**}**

y**++;**

i**++;**

**}**

mvwprintw**(**data\_win**,**y**,**0**,**"%S"**,**L"└───┴──────────┴──────────┴──────────┴──────────┴──────────┴──────────┴───────┘"**);**

**}**

/\* Отрисовка данных поиска по цеху \*/

void draw\_data\_find\_guild**()** **{**

mvwprintw**(**data\_win**,**11**,**30**,**"%S"**,**L"Номер цеха для поиска"**);**

mvwprintw**(**data\_win**,**12**,**30**,**"%S"**,**data\_find\_guild\_field**);**

/\* Перемещение курсора \*/

cursor\_x **=** 30**+**data\_find\_guild\_index**;**

cursor\_y **=** 12**;**

**}**

/\* Отрисовка данных поиска по имени \*/

void draw\_data\_find\_name**()** **{**

mvwprintw**(**data\_win**,**11**,**30**,**"%S"**,**L"Имя для поиска"**);**

mvwprintw**(**data\_win**,**12**,**30**,**"%S"**,**data\_find\_name\_field**);**

/\* Перемещение курсора \*/

cursor\_x **=** 30**+**data\_find\_name\_index**;**

cursor\_y **=** 12**;**

**}**

/\* Отрисовка данных формы \*/

void draw\_data\_form**()** **{**

form**::**form\_t**\*** form **=** form\_current**;**

form**::**field\_t**\*** field**;**

**for(**int i**=**0**;**i**<**form**->**length**;**i**++)** **{**

/\* Отрисовка поля \*/

field **=** form**->**field**[**i**];**

wattron**(**data\_win**,**COLOR\_PAIR**(**1**));**

mvwprintw**(**data\_win**,**field**->**y**,**field**->**x**,**"%S"**,**field**->**name**);**

wattron**(**data\_win**,**A\_REVERSE**);**

**for(**int j**=**0**;**j**<**field**->**length**;**j**++)** **{**

mvwprintw**(**data\_win**,**field**->**y**+**1**,**field**->**x**+**j**,**" "**);**

**}**

mvwprintw**(**data\_win**,**field**->**y**+**1**,**field**->**x**,**"%S"**,**field**->**data**);**

wattroff**(**data\_win**,**A\_REVERSE**);**

wattroff**(**data\_win**,**COLOR\_PAIR**(**1**));**

**}**

/\* Перемещение курсора \*/

cursor\_y **=** form**->**field**[**form**->**current\_field**]->**y **+** 1**;**

cursor\_x **=** form**->**field**[**form**->**current\_field**]->**x **+** form**->**field**[**form**->**current\_field**]->**index**;**

**}**

/\* Отрисовка сообщений \*/

void draw\_message**()** **{**

wbkgd**(**message\_win**,**COLOR\_PAIR**(**2**));**

wattron**(**message\_win**,**COLOR\_PAIR**(**2**));**

mvwprintw**(**message\_win**,**0**,**0**,**"%S"**,**L"╔════════════════════════════════════════════════╗"**);**

mvwprintw**(**message\_win**,**1**,**0**,**"%S"**,**L"║ ║"**);**

mvwprintw**(**message\_win**,**2**,**0**,**"%S"**,**L"╠════════════════════════════════════════════════╣"**);**

mvwprintw**(**message\_win**,**3**,**0**,**"%S"**,**L"║ ║"**);**

mvwprintw**(**message\_win**,**4**,**0**,**"%S"**,**L"║ ║"**);**

mvwprintw**(**message\_win**,**5**,**0**,**"%S"**,**L"║ ║"**);**

mvwprintw**(**message\_win**,**6**,**0**,**"%S"**,**L"╚════════════════════════════════════════════════╝"**);**

int l **=** wcslen**(**message\_current**.**title**);**

mvwprintw**(**message\_win**,**1**,(**50**-**l**)/**2**-**1**,**"%S"**,**message\_current**.**title**);**

l **=** wcslen**(**message\_current**.**text**);**

mvwprintw**(**message\_win**,**4**,(**50**-**l**)/**2**-**1**,**"%S"**,**message\_current**.**text**);**

mvwprintw**(**message\_win**,**5**,**7**,**"%S"**,**L"Для закрытия нажмите любую клавишу"**);**

wattroff**(**message\_win**,**COLOR\_PAIR**(**2**));**

wrefresh**(**message\_win**);**

**}**

**}**