Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерного проектирования

Кафедра инженерной психологии и эргономики

Дисциплина: Программируемые мобильные системы

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовой работе

на тему

**РАЗРАБОТКА ПРОГРАММЫ УЧЕТА ЗАРАБОТНОЙ ПЛАТЫ СОТРУДНИКОВ ПРЕДПРИЯТИЯ**

Выполнил: студент группы

Ф.И.О.

Проверил: Ф.И.О. преподавателя

Минск 2022

**СОДЕРЖАНИЕ**

1. Требования к программе.

2. Конструирование программы.

2.1 Разработка модульной структуры программы.

2.2 Выбор способа организации данных.

2.3 Разработка перечня пользовательских функций программы.

3. Разработка алгоритмов работы программы.

3.1 Алгоритм функции main.

3.2 Алгоритм функции enableRussianLanguage.

3.3 Алгоритм функции addUser.

4. Описание работы программы.

4.1 Авторизация.

4.2 Модуль администратора.

4.3 Модуль пользователя.

4.4 Исключительные ситуации.

Приложение.

**1 ТРЕБОВАНИЯ К ПРОГРАММЕ**

Необходимо разработать программу расчета заработной платы сотрудников предприятия.

Сведения о сотрудниках предприятия содержат:

- Ф.И.О. сотрудника;

- табельный номер;

- год;

- месяц;

- количество проработанных часов за месяц;

- почасовой тариф.

Рабочее время свыше 144 часов считается сверхурочным и считается в двойном размере.

Индивидуальное задание: рассчитать размер заработной платы каждого сотрудника за вычетом подоходного налога, который составляет 12%. Определить объем выплат конкретному сотруднику за требуемый период времени, который вводится с клавиатуры.

Также необходимо реализовать авторизацию для входа в систему, функционал администратора и пользователя.

* 1. Исходные данные

1. Тема: разработка программы расчета заработной платы сотрудников предприятия
2. Язык программирования: C++
3. Среда разработки: Microsoft Visual Studio
4. Вид приложения: консольное
5. Парадигма программирования: процедурная
6. Способ организации данных: структуры
7. Способ хранения данных: файлы
8. Каждая логически завершенная подзадача должна быть реализована в виде отдельной функции
9. Построение программного кода должно соответствовать соглашению о коде «C++ Code Convention»
10. К защите курсовой работы представляются: консольное приложение и пояснительная записка
11. Текст пояснительной записки оформляется в соответствии со стандартом предприятия СТП 01-2017
    1. Функциональные требования

Первый этап работы программы – авторизация – предоставление прав доступа.

В рамках данного этапа считываются данные из файла с учетными записями пользователей следующего вида:

- login;

- hashPassword;

- role;

- access.

Поле login хранит логин пользователя, hashPassword хранит зашифрованный пароль пользователя, role хранит права доступа, где 1 – администратор, 0 – пользователь, access хранит разрешение на вход в систему, где 1 – доступ разрешен, 0 – доступ запрещен.

Если файла с учетными записями пользователей не существует, то создается файл, где записываются данные администратора.

По соображениям безопасности, при вводе пароля на экране выводятся «звездочки».

Регистрация новых пользователей осуществляется администратором в режиме работы с учетными записями пользователе. При регистрации выполняется проверка на уникальность логина.

Также возможна регистрация пользователей при входе, после которой администратор должен подтвердить регистрацию в режиме редактирования учетных записей пользователей, путем изменения поля access.

Второй этап работы программы – работа с данными, которая становится доступной только после успешного входа в систему. Данные о сотрудниках хранятся в файле и имеют структуру:

- уникальный ID записи;

- фамилия сотрудника;

- имя сотрудника;

- отчество сотрудника;

- табельный номер;

- год;

- месяц;

- количество проработанных часов за месяц;

- почасовой тариф.

Для работы с данными предусмотрены два функциональных модуля: модуль администратора и модуль пользователя.

Модуль администратора содержит следующие подмодули:

1. Управление учетными записями пользователей:

- просмотр всех учетных записей пользователей

- добавление новой учетной записи пользователя

- редактирование учетной записи пользователя

- удаление учетной записи пользователя

Так как существует возможность регистрации при входе, администратор подтверждает аккаунт через подмодуль «Редактирование учетной записи пользователя».

Допускается создавать учетную запись пользователя для нескольких администраторов.

Администратору запрещено удалять или изменять права доступа и разрешение на вход в систему самому себе.

1. Работа с данными:
   1. Режим редактирования:

- просмотр всех записей;

- добавление новой записи;

- удаление записи;

- редактирование записи;

* 1. Режим обработки данных:

- выполнение индивидуального задания в рамках данной курсовой работы;

- поиск данных;

- сортировка;

Модуль пользователя содержит следующие подмодули:

1. Просмотр всех данных;
2. Режим обработки данных:

- выполнение индивидуального задания в рамках данной курсовой работы;

- поиск данных;

- сортировка;

Для реализации перечисленных модулей и подмодулей созданы меню с соответствующими пунктами.

В данной курсовой работе предусмотрены:

1. Исключительные ситуации:

- отсутствие файла с учетными записями пользователей;

- проверка на уникальность логина при регистрации и редактировании учетной записи пользователя;

- запрет на изменение некоторых полей и удаление учетной записи пользователя;

- введенные пользователем данные не соответствуют формату поля;

- введенные пользователем данные нелогичны (например, почасовой тариф меньше либо равен 0);

- файл с данными для чтения не существует;

- номер удаляемой записи выходит за пределы массива;

1. Возможность перемещения по программе;
2. Запрос на удаление при необратимых действиях: подтверждение при удалении данных;
3. Обратную связь с пользователем через сообщения.

1.3 Требования к программной реализации

1. Все переменные и константы должны иметь осмысленные имена в рамках тематики данной курсовой работы. Переменным рекомендуется присваивать имена, состоящие из букв нижнего регистра, для формирования составного имени используется «верблюжья нотация» (например, flagExit). Константам присваиваются имена, состоящие из букв верхнего регистра, и используется нижнее подчеркивание (например, FILE\_OF\_ACCOUNTS);
2. Имена функции должны быть осмысленными, начинаются с буквы нижнего регистра и строятся по принципу глагол+существительное (например, addAccount). Если функция выполняет проверку и возвращает результат типа bool, то ее название должно начинаться с глагола is (например, isNumberNumeric);
3. Не допускается использование оператора прерывания goto
4. Код не должен содержать неименованных числовых констант, неименованных строковых констант. Подобного рода информацию следует представлять как глобальные константы. По правилам качественного стиля программирования тексты всех информационных сообщений, выводимых пользователю в ответ на его действия, также оформляются как константы.
5. Код необходимо комментировать (как минимум в части объявления структур, массивов/векторов, прототипов функций, нетривиальной логики).
6. Код не должен дублироваться – для этого существуют функции!
7. Одна функция решает только одну задачу (например, не допускается в одной функции считывать данные из файла и выводить их на консоль – это две разные функции). При этом внутри функции возможен вызов других функций.
8. Выполнение операций чтения/записи в файл должно быть сведено к минимуму (т. е. после однократной выгрузки данных из файла в массив дальнейшая работа ведется с этим массивом, а не происходит многократное считывание данных из файла в каждой функции).
9. Следует избегать глубокой вложенности условных и циклических кон-струкций: вложенность блоков должна быть не более трех.
10. Следует избегать длинных функций: текст функции должен умещаться на один экран (размер текста не должен превышать 25–50 строк).
11. Следует выносить код логически независимых модулей в отдельные .cpp файлы и подключать их с помощью заголовочных .h файлов.

2 КОНСТРУИРОВАНИЕ

2.1 Разработка модульной структуры

2.2 Выбор способа организации данных

В качестве способа организации данных используются структуры следующего вида:

- Структура учётной записи пользователя:

- логин;

- зашифрованный пароль;

- права доступа;

- возможность входа в систему;

- Структура записи сотрудника:

- уникальный ID записи;

- фамилия сотрудника;

- имя сотрудника;

- отчество сотрудника;

- табельный номер;

- год;

- месяц;

- количество проработанных часов за месяц;

- почасовой тариф.

В качестве способа объединения входных данных используются глобальные динамические массивы структур.

2.3 Разработка перечня пользовательских функций

void enableRussianLanguage() – функция, которая подключает поддержку русского языка в консоли.

void login() – функция авторизации пользователей.

void exit() – функция завершения работы программы.

Блок функций, отвечающих за работу с файлом и динамическим массивом структур учетных записей пользователей:

- void cleanMemoryUsers() – функция, очищающая динамический массив структур учетных записей пользователей;

- void addAdmin\_fileWasntExist() – функция, добавляющая учетную запись администратора при создании файла учетных записей пользователей, если файл при запуске не существовал;

- void addUser() – функция, добавляющая новую пустую структуру в динамический массив структур учетных записей пользователей;

- void openUserFile() – функция, открывающая файл с учетными записями пользователями и передающая информацию из файла в динамический массив структур учетных записей пользователей;

- void createUserFile() – функция, создающая файл с учетными записями пользователей, если он отсутствует;

- void toUserFile() – функция, передающая информацию из динамического массива структур учетных записей пользователей в файл.

Блок функций-модулей пользователей разных прав доступа к системе:

- void adminMenu() – функция-модуль администратора;

- void userMenu() – функция-модуль пользователя.

Блок функций, отвечающих за работу с файлом и динамическим массивом структур сотрудников:

- void enterDataFilename() – функция, в которой пользователь вводит название файла с сотрудниками;

- void openDataFile() – функция, открывающая файл с сотрудниками и передающая информацию из файла в динамический массив структур сотрудников;

- void cleanMemoryData() – функция, очищающая динамический массив структур сотрудников;

- void createDataFile() – функция, создающая файл с сотрудниками, если он отсутствует;

- void addEmployee() – функция, добавляющая новую пустую структуру в динамический массив сотрудников;

- void toDataFile() – функция, передающая информацию из динамического массива структур сотрудников в файл.

Блок функций-подмодулей администратора:

- void userControlPanel() – функция-подмодуль управления учетными записями пользователей;

- void dataAdmin() – функция-подмодуль работы с данными о сотрудниках.

Блок функций-подмодуля управления учетными записями пользователей:

- void showAllUsers() – функция просмотра содержимого динамического массива структур учетных записей пользователей;

- void addUserInDatabase() – функция, добавляющая новую учетную запись пользователя в динамический массив структур;

- void editUser() – функция редактирования учетной записи пользователя в динамическом массиве структур;

- void deleteUser() – функция удаления учетной записи пользователя из динамического массива структур.

Блок функций-подмодуля управления учетными записями пользователй:

- void outputUsers() – функция вывода учетных записей пользователей с шапкой;

- void outputUser(int, bool) – функция вывода одной учётной записи пользователя;

- int getIndexOfUser() – функция, получающая индекс учетной записи пользователя в динамическом массиве структур;

- bool isAdminEditsThemself(int) – функция, которая проверяет: не редактирует ли администратор самого себя.

Блок функций-подмодуля работы с данными о сотрудниках:

- void modifyData() – функция-подмодуль режима редактирования данных;

- void dataProcessing() – функция-подмодуль режима обработки данных.

Блок функций-подмодуля режима обработки данных:

- void salary() – функция вывода зарплаты сотрудников;

- void linearSearch() – функция линейного поиска данных сотрудников;

- void sort() – функция сортировки данных сотрудников.

Блок функций-подмодуля режима редактирования данных о сотрудниках:

- void viewData() – функция просмотра содержимого динамического массива структур сотрудников;

- void addData() – функция добавления записи в динамический массив структур сотрудников;

- void deleteData() – функция удаления записи из динамического массива структур сотрудников;

- void editData() – функция редактирования записи в динамическом массиве структур сотрудников.

Блок функций-подмодулей режима редактирования данных о сотрудниках:

- void outputData() – функция вывода одной записи сотрудника;

- int getLocalYear() – функция получения года в системе;

- string toLower(string) – функция, опускающая регистр всего слова;

- int getFreeTabelNumber() – функция получения свободного табельного номера;

- int getFreeID() – функция получения свободного номера.

Блок функций-подмодулей режима обработки данных о сотрудниках:

- double getSalary(int, double) – функция вычисления зарплаты по количеству часов и почасовому тарифу;

- void sort\_getParametr(bool) – функция выбора критерия сортировки;

- void showCriteria() – функция вывода критериев сортировки;

- void bubbleSort(int) – функция пузырьковой сортировки;

- void quicksort(int, int, int) – функция быстрой сортировки;

- int getMiddleIndex(int, int, int) – вспомогательная функция быстрой сортировки;

- string getTempParametr(employee, int) – функция получения параметра из структуры сотрудника для поиска;

- int getID() – функция получения номера записи в динамическом массиве структур сотрудников;

- int getMaxID() – функция получения максимального номера записи в динамическом массиве структур сотрудников.

3 РАЗРАБОТКА АЛГОРИТМОВ РАБОТЫ ПРОГРАММЫ

3.1 Алгоритм функции main



3.2 Алгоритм функции enableRussianLanguage



3.3 Алгоритм функции addUser



4 ОПИСАНИЕ РАБОТЫ ПРОГРАММЫ

Описание работы программы подразумевает краткое словесное описание работы программы со скриншотами консоли

4.1 Авторизация

Когда запускается функция авторизации, запускается открытие файла с учетными записями пользователей в динамический массив структур. Затем выводится сообщение-предложение с вводом логина. Затем запускается бесконечный цикл, где пользователем вводится пароль. После ввода пароля, запускается цикл с нахождением совпадения логина в динамическом массиве структур учетных записей пользователей. Если программа находит совпадение, она сравнивает пароль, введённый пользователем, с паролем аккаунта из системы. Если совпадение есть, то программа выходит из бесконечного цикла, если совпадения нет, то выводится сообщение о неправильном пароле и пользователь снова вводит пароль. После бесконечного цикла, программа проверяет наличие аккаунта в динамическом массиве структур учетных записей пользователей, если учетная запись, а именно её индекс в динамическом массиве структур учетных записей пользователей не должен превышать общее количество учетных записей, существует в динамическом массиве, то в глобальные переменные присваивается логин и права доступа этой учетной записи, затем проверяется разрешение на доступ в систему, если доступ есть, то в зависимости от прав доступа запускается либо модуль администратора, либо модуль пользователя, если разрешение отсутствует, то выводится сообщение о закрытом доступе и предложение с входом в другой аккаунт. Если учетной записи пользователя не существует, а именно индекс учетной записи превышает количество учетных записей в динамическом массиве учетных записей пользователей, то выводится предложение с регистрацией аккаунта. Если пользователь желает зарегистрироваться, то создаётся новая пустая структура, в которую переносится логин, пароль, и права доступа, которые пользователь желает иметь. После выбора, файл с учетными записями обновляется. Если пользователь не желает зарегистрироваться, то программа завершает свою работу.

4.2 Модуль администратора

При запуске модуля администратора, запускается бесконечный цикл – меню, в котором пользователь может выбрать запуск одного из двух подмодулей, выход из системы или завершение работы. При выборе одного из модуля, из файла считываются данные в динамический массив структур, логическая переменная editsInStructures, которая отвечает за изменения в динамическом массиве структур, и запускается подмодуль, который выбрал пользователь. Если пользователь желает выйти из системы, запускается функция авторизации. Если пользователь желает завершить работу программы, запускается функции завершения работы программы. Если пользователь ввёл неправильный номер, то выводится сообщение о неправильном номере и пользователю снова предлагается выбрать пункт меню. Если пользователь ввёл значение, отличное от int, то данное исключение обрабатывается в функции inputInt библиотеки input.h.

4.3 Модуль пользователя

Модуль пользователя работает аналогично, как и модуль администратора, но отличается другим набором меню: просмотр и обработка данных их файла с сотрудниками, открытие другого файла с сотрудниками, выход из системы и завершение работы.

4.4 Исключительные ситуации

В данной курсовой работы предусмотрены различные исключительные ситуации:

- отсутствие файла с учётными записями пользователей;

- проверка на уникальность логина при регистрации и редактировании учетной записи пользователя;

- если администратор вошёл в систему, то ему запрещается изменять себе некоторые поля или удалять свою учетную запись;

- введенные пользователем данные не соответствуют формату поля;

- введенные пользователем данные нелогичны (например, почасовой тариф меньше либо равен 0);

- файл с данными для чтения не существует;

- номер удаляемой записи выходит за пределы массива.

ПРИЛОЖЕНИЕ

листинг кода с комментариями (приводится ВЕСЬ код с авторским форматированием и комментариями; допускается применение в листинге кода шрифта 12 пунктов и выше