Министерство образования Республики Беларусь

Учреждение образования

«Гомельский государственный университет   
имени Франциска Скорины»

Факультет физики и

информационных технологий

Кафедра общей физики

**Разработка программы начисления**

**пособий по уходу за ребёнком**

Курсовая работа

по учебной дисциплине

"Технология разработки программного обеспечения"

Исполнитель

студент группы МС-12 Исаченко Ю. С.

Научный руководитель Грищенко В. В.

Гомель 2022

**РЕФЕРАТ**

**Курсовой проект:** 30 страниц, 45 рисунков.

**Ключевые слова:** Си, программирование, разработка программы.

**Цель курсовой работы:** Получение теоретических знаний и практических навыков по конструированию и реализации программ на языке высокого уровня в рамках процедурной парадигмы.

**Метод исследования**: теоретический метод, эксперимент.

**Задачи курсовой работы:**

1. Анализ предметной области;
2. Разработка сценария работы программы;
3. Анализ исходных данных для проектирования программы;
4. Разработка программы;
5. Тестирование работы программы;

**Вывод:** В ходе данного курсового проекта была спроектирована и разработана программа для расчета пособий по уходу за ребёнком

Содержание

[**Введение** 4](#_Toc122991897)

[**1.Язык СИ.** 5](#_Toc122991898)

[**1.1 Характерные черты языка** 6](#_Toc122991899)

[**1.2 Процедурная парадигма.** 8](#_Toc122991900)

[**2. Разработка программы** 10](#_Toc122991901)

[2.1 Условие 10](#_Toc122991902)

[2.2 Требования 10](#_Toc122991903)

[**3. Код программы** 12](#_Toc122991904)

[3.1 Библиотеки, стандартные значения, структуры, глобальные перменные 12](#_Toc122991905)

[3.1 Функция main 13](#_Toc122991906)

[3.3 Функция authorization 14](#_Toc122991907)

[3.4 Функция array\_nulling 15](#_Toc122991908)

[3.5 Функция userPanel 16](#_Toc122991909)

[3.6 Функция adminPanel 17](#_Toc122991910)

[3.7 Функция outputInformationAboutEmployeesForUser 18](#_Toc122991911)

[3.8 Функция outputInformationAboutEmployeesForAdmin 18](#_Toc122991912)

[3.9 Функция addNewEmployee 19](#_Toc122991913)

[3.10 Функция benefitCalculation 20](#_Toc122991914)

[3.11 Функция editInformationAboutEmployees 21](#_Toc122991915)

[3.12 Функция editInformationAboutChildren 22](#_Toc122991916)

[3.13 Функция outputChildren 22](#_Toc122991917)

[3.14 Функция editeInformationAboutCurrentChildren 23](#_Toc122991918)

[3.15 Функция addNewChild 24](#_Toc122991919)

[3.16 Функция deleteChild 24](#_Toc122991920)

[3.17 Функция deleteEmployee 25](#_Toc122991921)

[3.18 Функция outputUsers 27](#_Toc122991922)

[3.19 Функция addNewUser 27](#_Toc122991923)

[3.20 Функция editUsers 28](#_Toc122991924)

[3.21 Функция deleteUser 29](#_Toc122991925)

[**Заключение** 30](#_Toc122991926)

[Список использованных источников: 31](#_Toc122991927)

[Приложение 31](#_Toc122991928)

# 

# **Введение**

Целью курсовой работы является получение студентами теоретических знаний и практических навыков по конструированию и реализации программ на языке высокого уровня в рамках процедурной парадигмы.

На уровне конструирования задачей курсовой работы является освоение технологии нисходящего проектирования и принципов методологии

структурного программирования. Результат этапа конструирования

представляется в форме графических схем алгоритмов.

На уровне программирования задачей курсовой работы является

реализация графических схем алгоритмов на языке высокого уровня с

соблюдением правил соглашения о коде (Code Convention). Результат этапа

программирования представляется в форме проекта в среде разработки

приложений на языке высокого уровня (например, Microsoft Visual Studio при использовании языка программирования С).

# **1.Язык СИ.**

Язык программирования Си — универсальный язык программирования, который завоевал особую популярность у программистов, благодаря сочетанию возможностей языков программирования высокого и низкого уровней. Большинство программистов предпочитают использовать язык Си для серьезных разработок потому, что их привлекают такие особенности языка, как свобода выражения мыслей, мобильность и чрезвычайная доступность.

Язык Си даёт возможность программисту осуществлять непосредственный доступ к ячейкам памяти и регистрам компьютера, требуя при этом знания особенностей функционирования ЭВМ. В этом Си схож с языком низкого уровня — ассемблером, хотя на самом деле он представляет собой гораздо более мощное средство решения трудных задач и создания сложных программных систем.

Язык Си был разработан американцем Деннисом Ритчи в исследовательском центре Computer Science Research Center of Bell Laboratories корпорации AT&T в 1972 г. Первоначальная реализация Си была выполнена на ЭВМ PDP-11 фирмы DEC для создания операционной системы UNIX. Позже он был перенесен в среду многих операционных систем и существует независимо от любой из них. Программы, написанные на языке Си, как правило, можно перенести в любую другую операционную систему или на другой компьютер либо с минимальными изменениями, либо вовсе без них.

Язык Си также используется при составлении программ для микроконтроллеров.

Язык Си оказал существенное влияние на развитие индустрии программного обеспечения, а его синтаксис стал основой для таких языков программирования как C++, C#, Java, PHP и др.

## **1.1 Характерные черты языка**

Алфавит языка Си включает:

- прописные и строчные буквы латинского алфавита, а также знак подчеркивания (код ASCII 95);

- арабские цифры от 0 до 9;

- специальные символы:

+(плюс) –(минус) \*(звездочка) /(дробная черта) =(равно) >(больше) <(меньше) ;(точка с запятой) &(амперсант) [ ](квадратные скобки) { }(фигурные скобки) ()(круглые скобки) \_(знак подчеркивания) .(точка) ,(запятая) :(двоето­чие) #("решетка") %(процент) ~(поразрядное отрицание) ?(знак вопроса) !(восклица­­тельный знак) \(обратный слеш).

- пробельные (разделительные) символы: пробел, символы табуляции, перевода строки, возврата каретки.

Из символов алфавита формируются лексемы языка – минимальные значимые единицы текста в программе:

- идентификаторы;

- ключевые (зарезервированные) слова;

- знаки операций;

- константы;

- разделители (скобки, точка, запятая, пробельные символы).

Границы лексем определяются другими лексемами, такими, как разделители или знаки операций, а также комментариями.

Идентификатор (в дальнейшем, для краткости - ID) – это имя программного объекта (константы, переменной, метки, типа, функции, модуля, поля в структуре). В идентификаторе могут использоваться латинские буквы, цифры и знак подчеркивания; первым символом ID может быть буква или знак подчеркивания, но не цифра; пробелы внутри ID не допускаются.

Длина иденти­фикатора определяется реализацией (версией) транслятора Cи и редактора связей (компоновщика). Современная тенденция - снятие ограничений длины идентификатора.

При именовании объектов следует придерживаться общепринятых соглашений:

- ID переменной обычно пишется строчными буквами, например index (для сравнения: Index – это ID типа или функции, а INDEX – константа);

- идентификатор должен нести какой-либо смысл, поясняя назначение объекта в программе, например birth\_date (день рождения) или sum (сумма);

- если ID состоит из нескольких слов, как, например birth\_date, то принято либо разделять слова символом подчеркивания (birth\_date), либо писать каждое следующее слово с большой буквы (birthDate).

Разделители идентификаторов объектов:

- пробелы;

- символы табуляции, перевода строки и страницы;

- комментарии (играют роль пробелов).

Наличие разделителей не влияет на работу программы.

В Си прописные и строчные буквы – различные символы. Идентификаторы Name, NAME, name – различные объекты.

Ключевые (зарезервированные) слова не могут быть использованы в качестве идентификаторов.

Когда в программе встречается некоторое число, например 1, то это число называется литералом или литеральной константой. Константой, потому что мы не можем изменить его значение, и литералом, потому что буквально передает свое значение (от латинского literal – буквальный).

Литерал является неадресуемой величиной: хотя реально он, конечно, хранится в памяти машины, нет никакого способа узнать его адрес. Каждый литерал имеет определенный тип.

Еще один базовый элемент языка программирования – комментарий, – не является лексемой. Внутри комментария можно использовать любые допустимые на данном компьютере символы, а не только символы из алфавита языка программирования, поскольку компилятор комментарии игнорирует.

В Си комментарии ограничиваются парами символов /\* и \*/, а в С++ был введен вариант комментария, который начинается символами // и заканчивается символом перехода на новую строку.

## **1.2 Процедурная парадигма.**

**Парадигма** — это «набор предположений, концепций, ценностей и практик, которые представляют собой способ рассмотрения реальности для сообщества, которое разделяет их, особенно в интеллектуальной дисциплине ».

**Процедурное программирование –** это парадигма, которая использует линейный нисходящий подход и рассматривает данные и процедуры как два разных объекта. Основываясь на концепции вызова процедуры, процедурное программирование делит программу на процедуры, которые также известны как процедуры или функции, просто содержащие последовательность шагов, которые необходимо выполнить.

**Ключевые особенности процедурного программирования:**

* Предопределенные функции. Предопределенная функция — это обычно инструкция, идентифицируемая именем. Обычно предопределенные функции встроены в языки программирования более высокого уровня , но они получены из библиотеки или реестра, а не из программы. Одним из примеров предопределенной функции является charAt (), которая ищет позицию символа в строке.
* Локальная переменная. Локальная переменная — это переменная, которая объявлена ​​в основной структуре метода и ограничена локальной областью действия, которую она задает. Локальная переменная может использоваться только в том методе, в котором она определена, и, если она будет использоваться вне определенного метода, код перестанет работать.
* Глобальная переменная. Глобальная переменная — это переменная, которая объявляется вне любой другой функции, определенной в коде. Благодаря этому глобальные переменные могут использоваться во всех функциях, в отличие от локальной переменной.
* Модульность. Модульность — это когда две разные системы имеют под рукой две разные задачи, но сгруппированы вместе, чтобы сначала выполнить более крупную задачу. В этом случае каждая группа систем будет выполнять свои собственные задачи один за другим, пока все задачи не будут выполнены.
* Передача параметров. Передача параметров — это механизм, используемый для передачи параметров в функции, подпрограммы или процедуры. Передача параметров может быть выполнена через «передача по значению», «передача по ссылке», «передача по результату», «передача по значению-результату» и «передача по имени».

**Преимущества и недостатки процедурного программирования:**

1. **Преимущества**

* Процедурное программирование отлично подходит для программирования общего назначения
* Закодированная простота наряду с простотой реализации компиляторов и интерпретаторов
* Большое разнообразие книг и онлайн-материалов курсов по проверенным алгоритмам, облегчающих обучение в процессе
* Исходный код переносим, ​​поэтому его можно использовать и для другого процессора
* Код может быть повторно использован в разных частях программы, без необходимости копировать его
* Благодаря методике процедурного программирования требования к памяти также сокращаются
* Ход программы можно легко отследить

1. **Недостатки**

* Код программы труднее писать, когда используется процедурное программирование
* Процедурный код часто не может быть использован повторно, что может привести к необходимости воссоздания кода, если это необходимо для использования в другом приложении
* Сложно общаться с реальными объектами
* Важное значение придается операции, а не данным, что может создавать проблемы в некоторых случаях, связанных с данными
* Данные открыты для всей программы, что делает их не очень безопасными

# **2. Разработка программы**

# Условие

Разработка программы начисления пособий по уходу за ребенком. Сведения о детях сотрудниц компании содержат следующую информацию: ФИО сотрудницы, дата рождения ребенка, ФИО ребенка. Для каждого ребенка сотрудницы создается отдельная запись указанного вида.

**Индивидуальное задание:** Для каждой сотрудницы вывести количество детей, а также рассчитать общую сумму полагающихся пособий. Пособие начисляется для детей в возрасте до трех лет. Для первого ребенка в семье базовое пособие составляет X (вводится с клавиатуры), для второго – 1,25X, для третьего и последующих – 1,5X.

# 2.2 Требования

Первым этапом работы программы является авторизация –

предоставление прав доступа. В рамках данного этапа необходимо считать

данные из файла с учетными записями пользователей следующего вида:

login; password; role (данное поле служит для разделения в правах

администраторов и пользователей).

После ввода пользователем своих персональных данных (логина и

пароля) и сверки со считанной из файла информацией необходимо

предусмотреть возможность входа: в качестве администратора; в качестве

пользователя.

Регистрация новых пользователей осуществляется администратором в режиме работы с учетными записями пользователей (т. е. администратор сам создает для пользователей аккаунты).

Вторым этапом работы программы является собственно работа с

данными, которая становится доступной только после прохождения

авторизации. Данные хранятся в отдельном файле и имеют вид, описанный подробно в каждом варианте к курсовой работе.

Для работы с данными должны быть предусмотрены два

функциональных модуля: модуль администратора и модуль пользователя.

Модуль администратора включает следующие подмодули (с указанием функциональных возможностей):

1. Управление учетными записями пользователей: просмотр всех

учетных записей; добавление новой учетной записи; редактирование учетной записи; удаление учетной записи.

2. Работа с данными:

а) режим редактирования: просмотр всех данных; добавление

новой записи; удаление записи; редактирование записи;

б) режим обработки данных: выполнение индивидуального

задания; поиск данных; сортировка.

Модуль пользователя включает подмодуль работы с данными со

следующими функциональными возможностями: просмотр всех данных;

выполнение индивидуального задания; поиск данных; сортировка.

Для реализации перечисленных модулей/подмодулей необходимо

создавать меню с соответствующими пунктами.

В курсовой работе необходимо предусмотреть:

1. Обработку исключительных ситуаций: введенные пользователем

данные не соответствуют формату поля (например, символы в числовом

поле); введенные пользователем данные нелогичны (например,

отрицательная цена товара); файл с данными для чтения не существует;

ничего не найдено по результатам поиска; номер удаляемой записи выходит

за пределы массива/вектора.

2. Возможность возврата назад (навигация).

3. Запрос на выполнение необратимых действий, а именно

подтверждение удаления вида «Вы действительно хотите удалить файл

(запись)?».

4. Обратную связь с пользователем, например, вывод сообщения об

успешности удаления/редактирования записи и т. д.

# **3. Код программы**

# 3.1 Библиотеки, стандартные значения, структуры, глобальные перменные

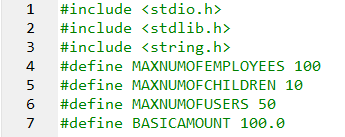


Рисунок 1 – Подключение библиотек и объявление стандартных значений (Максимальное количество сотрудников, максимальное количество детей у сотрудника, максимальное количество пользователей, сумма базового пособия)

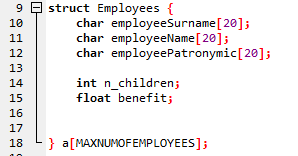


Рисунок 2 – Структура информации о сотрудниках, включающая в себя ФИО сотрудника, количество детей и сумму пособия.

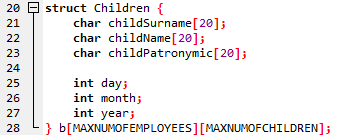


Рисунок 3 – Структура информации о детях сотрудников, включающая в себя ФИО, дату рождения ребёнка.

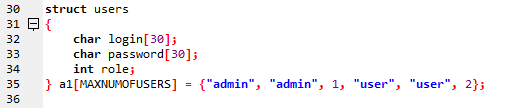


Рисунок 4 – Структура информации о пользователях, включающая в себя логин, пароль и роль (администратор или обычный пользователь).

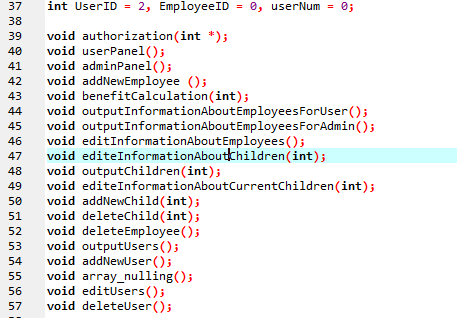


Рисунок 5 – Глобальные переменные и прототипы функций.

# 3.1 Функция main

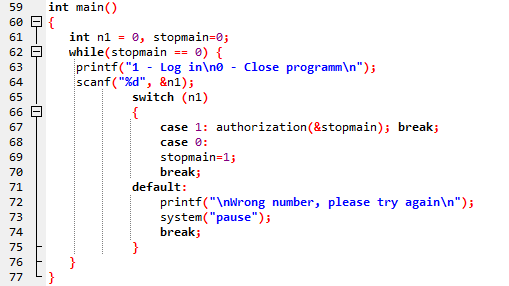


Рисунок 6 – Основная часть программы.

Данная функция является началом выполнения программы. Выводит первоначальные меню, в котором можно перейти к функции авторизации или выйти из программы.



Рисунок 7 – Первоначальное меню программы

# 3.3 Функция authorization

# 

Рисунок 8 – Функция авторизации.

Функция предлагает пользователю ввести логин и пароль, после чего введенные данные сверяет с данными в базе данных пользователей.

Если введенный логин и пароль совпадают с логином и паролем в базе данных, проверяется роль данной учётной записи. Если роль равна 1, то выполняется вход в аккаунт администратора. Если роль равна 0, то выполняется вход в аккаунт пользователя.

Если введенный логин и пароль не совпадают с логином и паролем в базе данных, то пользователю предлагается попробовать ввести логин/пароль ещё раз или выйти из программы.

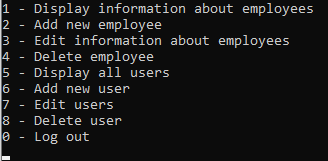


Рисунок 9 – Успешный вход в качестве администратора



Рисунок 10 – Успешный вход в качестве пользователя

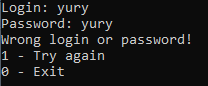


Рисунок 11 – Введенный логин/пароль не совпадает с логином/паролем в базе данных

# 3.4 Функция array\_nulling

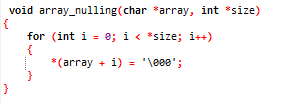


Рисунок 12 – Функция зануления массива.

Эта функция предназначена для обнуления(очищения) массива логина и пароля, которые ввёл пользователь, в функции авторизации. Необходима для стабильной работы функции авторизации.

# 3.5 Функция userPanel

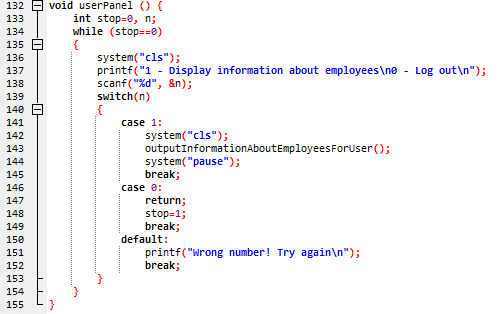


Рисунок 13 – Функция меню стандартного пользователя.

Данная функция вызывается, если будет выполнен успешный вход с ролью пользователя. Предоставляет возможность просмотреть информацию о сотрудниках (ФИО, количество детей и размер пособия) или выйти из программы.



Рисунок 14 – Меню стандартного пользователя

# 3.6 Функция adminPanel



Рисунок 15 - Функция меню администратора.

Данная функция вызывается, если будет выполнен успешный вход с ролью администратора. Предоставляет возможность работать с данными о сотрудниках и о пользователях. А именно, просмотреть стандартную информацию о сотрудниках (ФИО, количество детей и размер пособия), а также подробную информацию о детях каждого сотрудника, добавить нового сотрудника, редактировать информацию о сотруднике, удалить сотрудника. Также позволяет просмотреть информацию о пользователях(Логин, пароль, роль), добавить нового пользователя, редактировать данные о пользователях, удалить пользователя.

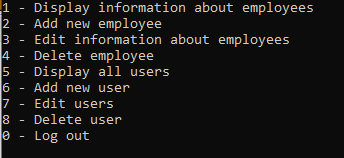


Рисунок 16 – Меню администратора

# 3.7 Функция outputInformationAboutEmployeesForUser

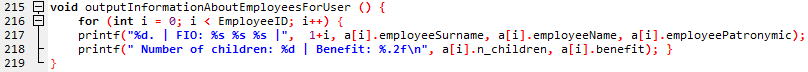


Рисунок 17 – Функция вывода информации о сотрудниках для пользователя.

Данная функция выводит стандартную информацию о каждом сотруднике (ФИО, количество детей, размер пособия).



Рисунок 18 – Вывод информации о сотрудниках для пользователя

# 3.8 Функция outputInformationAboutEmployeesForAdmin

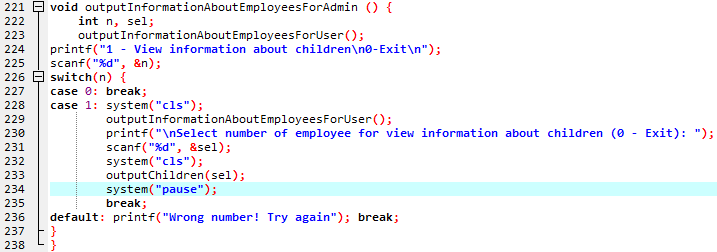


Рисунок 19 – Функция вывода информации о сотрудниках для администратора.

Данная функция вызывает функцию вывода стандартной информации о сотрудниках (ФИО, количество детей, размер пособия), а также позволяет просмотреть подробную информацию о детях сотрудников.

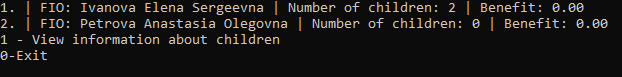


Рисунок 20 – Вывод информации о сотрудниках для администратора.

# 3.9 Функция addNewEmployee

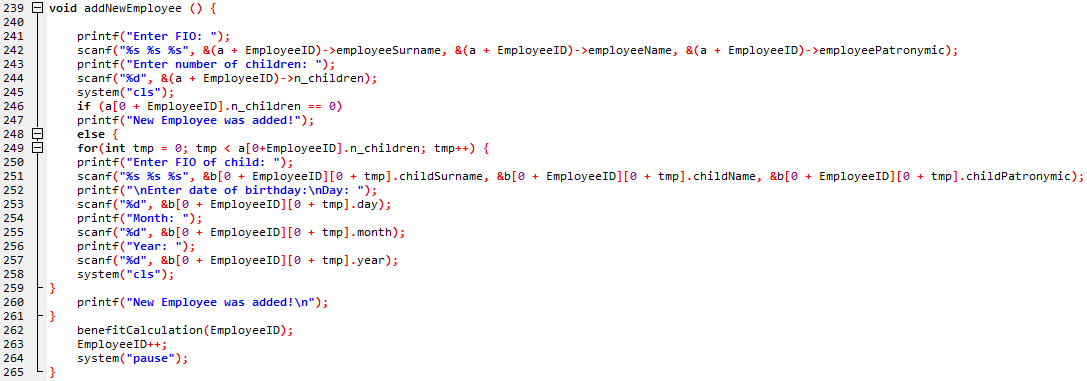


Рисунок 20 – Функция добавления нового сотрудника.

Данная функция позволяет добавить нового сотрудника в базу сотрудников. Просит пользователя ввести ФИО сотрудника и количество детей. Если количество детей равно нулю, завершает добавления нового пользователя. Если количество детей больше нуля, просит ввести ФИО и дату рождения каждого ребенка поочередно. После добавления информации о сотруднике, автоматически высчитывает сумму пособия.



Рисунок 21 – Результат добавления нового сотрудника

# 3.10 Функция benefitCalculation

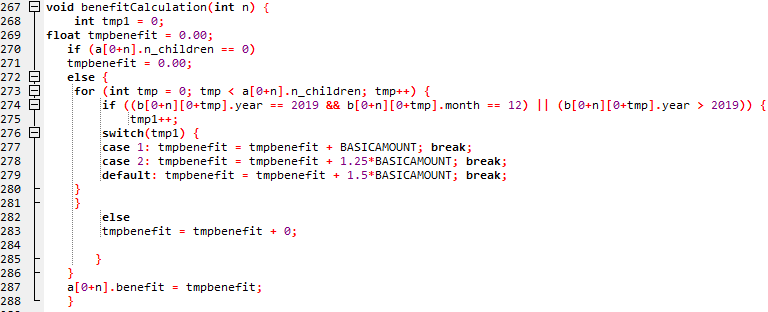


Рисунок 22 – Функция расчёта пособия.

Данная функция рассчитывает сумму пособия для сотрудника. На вход функции поступает номер сотрудника, пособие которого нужно рассчитать. Функция проверяет возраст каждого ребёнка сотрудника на декабрь 2022 года. Если ребёнок старше 3-х лет, пособие за него равно нулю. Если ребёнок младше 3-х лет, за него начисляется пособие в соответствие с изначальным требованием к программе. За первого ребёнка, младше 3-х лет, сумма пособия равна стандартному значению(X), за второго ребёнка, младше 3-х лет, сумма пособия равна 1.25\*X, за третьего и последующих 1.5\*X.

 Рисунок 23 – Вывод рассчитанной суммы пособия при выводе информации о сотрудниках.

# 3.11 Функция editInformationAboutEmployees

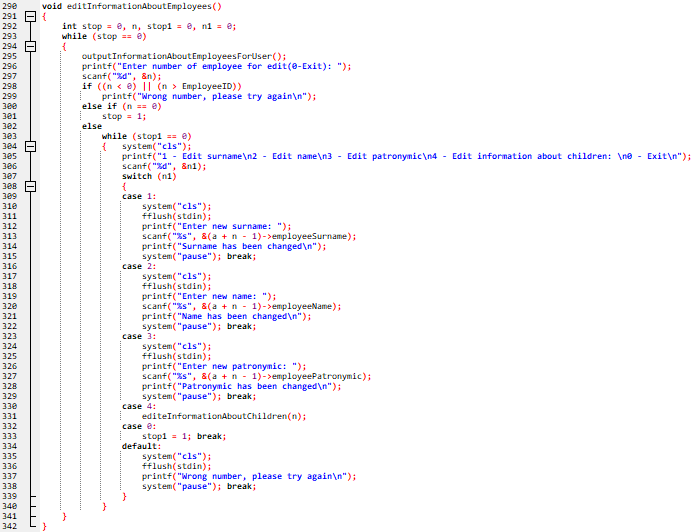


Рисунок 24 – Функция редактирования информации о сотруднике

Данная функция позволяет редактировать информацию о сотруднике. Выводит список сотрудников и позволяет выбрать номер сотрудника из списка. После выбора предлагает возможность редактировать ФИО, информацию о детях, выход из функции.



Рисунок 25 – Результат редактирования фамилии

# 3.12 Функция editInformationAboutChildren

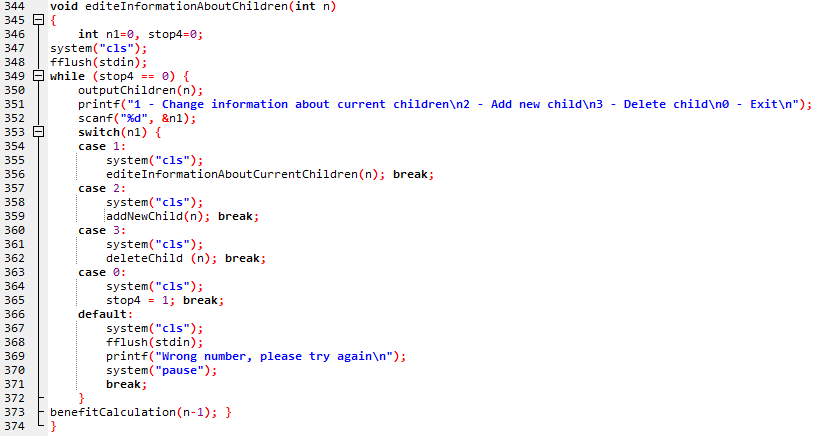


Рисунок 26 – Функция редактирования информации о детях сотрудника.

Данная функция вызывается из функции редактирования информации о сотруднике. На вход поступает номер сотрудника. Выводит список детей сотрудника, предлагает выбрать номер ребёнка для редактирования. Есть возможность редактировать информацию о нынешних детях сотрудника, добавить нового ребёнка, удалить ребёнка. Вызывает необходимую функцию, в зависимости от выбора пользователя. После окончания редактирования автоматически пересчитывает размер пособия.

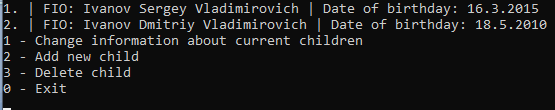


Рисунок 27 – Меню редактирования информации о детях сотрудника.

# 3.13 Функция outputChildren

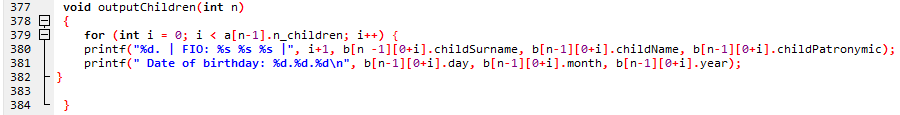


Рисунок 28 – Функция вывода информации о детях сотрудника

Данная функция выводит информацию о детях сотрудника (ФИО, дату рождения).



Рисунок 29 – Вывод информации о детях сотрудника

# 3.14 Функция editeInformationAboutCurrentChildren

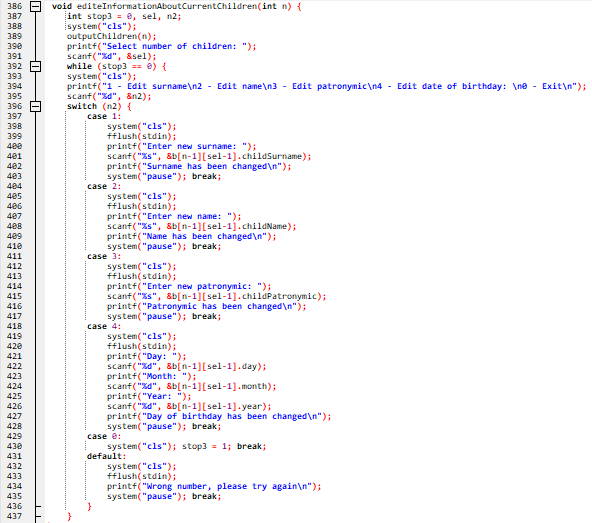


Рисунок 30 – Функция редактирования информации о нынешних детях сотрудника

Данная функция предназначена для редактирования информации о нынешних детях сотрудника. На вход поступает номер сотрудника. Выводит информацию о детях данного сотрудника. Есть возможность редактировать ФИО, дату рождения ребёнка.



Рисунок 31 – Результат редактирования имени ребёнка сотрудника.

# 3.15 Функция addNewChild

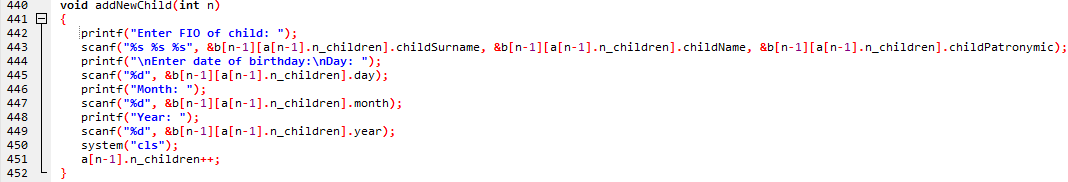


Рисунок 32 – Функция добавления нового ребёнка.

Данная функция добавляет ребёнка сотрудника. На вход поступает номер сотрудника. Программа просит пользователя ввести ФИО, дату рождения нового ребёнка. После завершения увеличивает количество детей сотрудника, выводит обновленный список детей сотрудника.

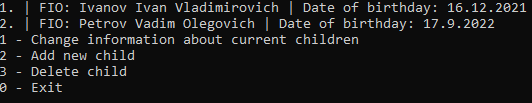


Рисунок 33 – Результат добавления нового ребёнка.

# 3.16 Функция deleteChild

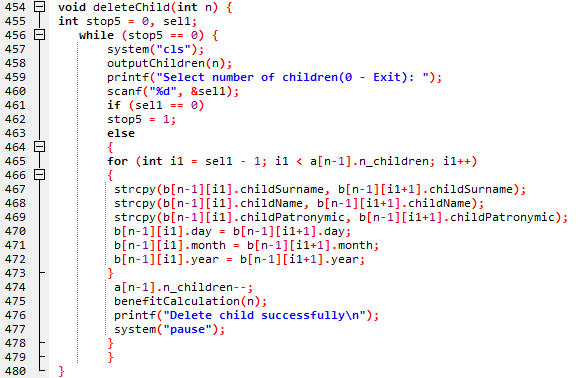


Рисунок 34 – Функция удаления ребёнка.

Данная функция удаляет ребёнка сотрудника. На вход поступает номер сотрудника. Выводит список детей сотрудника и предлагает выбрать номер ребёнка для удаления. После удаления уменьшает количество детей сотрудника. В базе информации о детях сотрудника, на место удаленного становится следующий за ним.

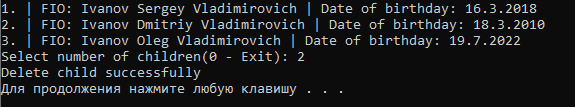


Рисунок 35 – Результат удаления ребёнка сотрудника.

# 3.17 Функция deleteEmployee



Рисунок 36 – Функция удаления сотрудника.

Данная функция удаляет сотрудника. Выводит список сотрудников и предлагает выбрать номер сотрудника для удаления. При удалении сотрудника удаляется и информация о его детях. В базе сотрудников на его место становится следующий за ним.

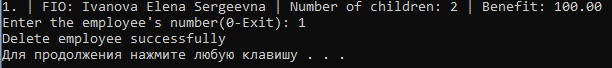


Рисунок 37 – Результат удаления сотрудника

# 3.18 Функция outputUsers

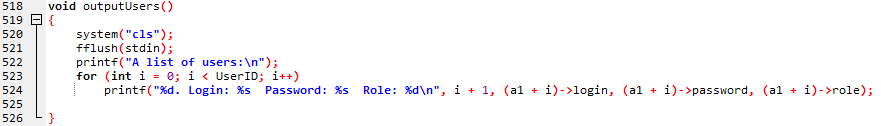


Рисунок 38 – Функция вывода информации о пользователях.

Данная функция выводит список пользователей и информацию о них (Логин, пароль, роль).

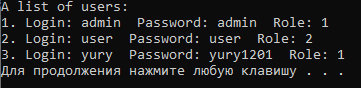


Рисунок 39 – Вывод информации о пользователях

# 3.19 Функция addNewUser

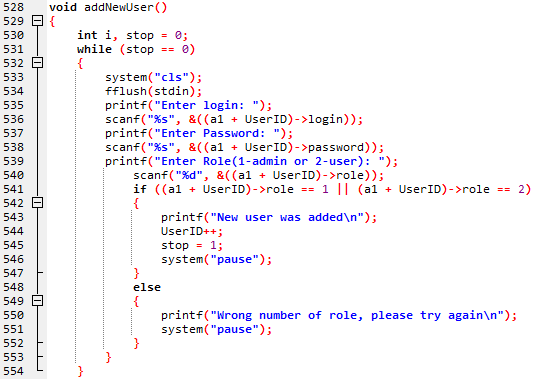


Рисунок 40 – Функция добавления нового пользователя

Данная функция предназначена для добавления нового пользователя. Просит ввести логин, пароль и роль нового пользователя.

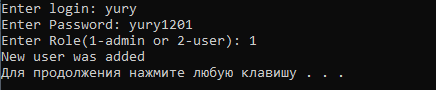


Рисунок 41 – Результат добавления пользователя

# 3.20 Функция editUsers

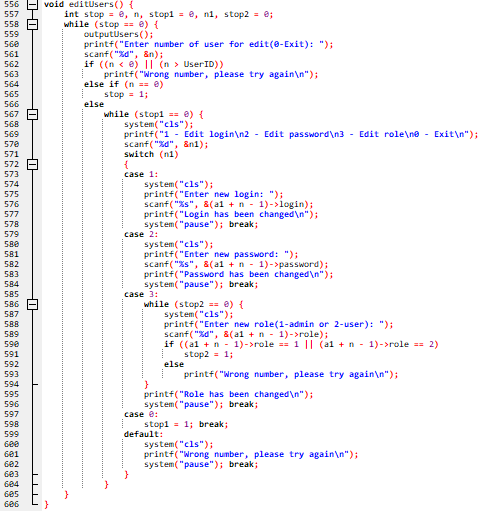


Рисунок 42 – Функция редактирования информации о пользователе

Данная функция предназначена для редактирования информации о пользователе. Выводит список сотрудников и предлагает выбрать номер пользователя, либо выйти из функции. После выбора номера пользователя предлагает редактировать логин, пароль, роль либо выйти из функции.



Рисунок 43 – Результат редактирования логина пользователя

# 3.21 Функция deleteUser

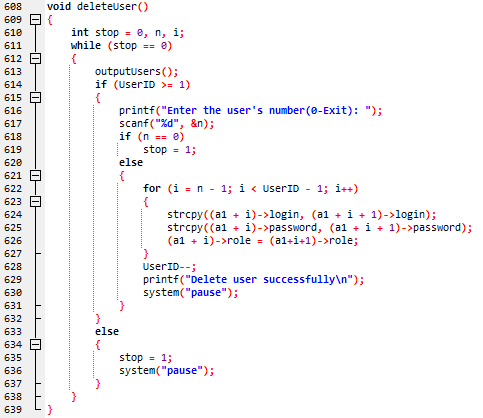


Рисунок 44 – Функция удаления пользователя

Данная функция удаляет пользователя. Вывод список пользователей и предлагает выбрать номер пользователя для удаления либо выйти из функции.

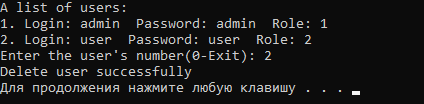


Рисунок 45 – Результат удаления пользователя.

# **Заключение**

В результате выполнения курсовой работы было разработано приложение для начисления пособий по уходу за ребёнком, которое имеет функцию авторизации, модули для двух типов учётных записей, имеет функции регистрации новых пользователей администратором, удаления пользователей администратором, редактирования учётных записей пользователей, добавление новых сотрудников администратором, удаления сотрудников администратором, редактирования информации о сотрудниках администратором, автоматического расчёта суммы пособия, а также содержит возможность навигации по программе, имеет обратную связь с пользователем. Были получены теоретические знания и практические навыки по конструированию и реализации программ на языке высокого уровня в рамках процедурной парадигмы.

# Список использованных источников:

1. C – Википедия [Электронный ресурс]. – 2022. – Режим доступа: https://ru.wikipedia.org/wiki/Си\_(язык\_программирования). – Дата доступа: 26.12.2022.
2. Язык Си – ProgCPP [Электронный ресурс]. – 2019. – Режим доступа: https://prog-cpp.ru/c/. – Дата доступа: 26.12.2022.
3. Алфавит языка Си – Studfile [Электронный ресурс]. – 2022. – Режим доступа: https://studfile.net/preview/5974500/. – Дата доступа: 26.12.2022.
4. Что такое процедурное программирование? – Edoodoo [Электронный ресурс]. – 2022. – Режим доступа: http://edoodoo.ru/chto-takoe-proczedurnoe-programmirovanie/. – Дата доступа: 26.12.2022.

# Приложение

Листинг программы

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAXNUMOFEMPLOYEES 100

#define MAXNUMOFCHILDREN 10

#define MAXNUMOFUSERS 50

#define BASICAMOUNT 100.0

struct Employees {

char employeeSurname[20];

char employeeName[20];

char employeePatronymic[20];

int n\_children;

float benefit;

} a[MAXNUMOFEMPLOYEES];

struct Children {

char childSurname[20];

char childName[20];

char childPatronymic[20];

int day;

int month;

int year;

} b[MAXNUMOFEMPLOYEES][MAXNUMOFCHILDREN];

struct users

{

char login[30];

char password[30];

int role;

} a1[MAXNUMOFUSERS] = {"admin", "admin", 1, "user", "user", 2};

int UserID = 2, EmployeeID = 0, userNum = 0;

void authorization(int \*);

void userPanel();

void adminPanel();

void addNewEmployee ();

void benefitCalculation(int);

void outputInformationAboutEmployeesForUser();

void outputInformationAboutEmployeesForAdmin();

void editInformationAboutEmployees();

void editeInformationAboutChildren(int);

void outputChildren(int);

void editeInformationAboutCurrentChildren(int);

void addNewChild(int);

void deleteChild(int);

void deleteEmployee();

void outputUsers();

void addNewUser();

void array\_nulling();

void editUsers();

void deleteUser();

int main()

{

int n1 = 0, stopmain=0;

while(stopmain == 0) {

printf("1 - Log in\n0 - Close programm\n");

scanf("%d", &n1);

switch (n1)

{

case 1: authorization(&stopmain); break;

case 0:

stopmain=1;

break;

default:

printf("\nWrong number, please try again\n");

system("pause");

break;

}

}

}

void authorization(int \*stopmain) {

int n = 0;

int stop = 0, stop1 = 0;

int flag = 0;

char login[30];

char password[30];

int maxNumofLetters = 30;

array\_nulling(login, &maxNumofLetters);

array\_nulling(password, &maxNumofLetters);

system("cls");

while (stop == 0) {

printf("Login: ");

scanf("%s", &login);

printf("Password: ");

scanf("%s", &password);

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

if (strcmp(login, a1[i].login) == 0 && strcmp(password, a1[i].password) == 0) {

if(a1[0+i].role == 1) {

adminPanel();

return; }

else

userPanel();

return;

}

else

flag = 1; }

if (flag == 1) {

while (stop1 == 0) {

printf ("Wrong login or password!\n1 - Try again\n0 - Exit\n");

scanf("%d", &n);

switch(n) {

case 1:

authorization(stopmain); break;

case 0:

stop = 1; stop1=1; flag = 0; \*stopmain = 1; break;

default: printf("\nWrong number, please try again\n");

}

}

}

}

}

void array\_nulling(char \*array, int \*size)

{

for (int i = 0; i < \*size; i++)

{

\*(array + i) = '\000';

}

}

void userPanel () {

int stop=0, n;

while (stop==0)

{

system("cls");

printf("1 - Display information about employees\n0 - Log out\n");

scanf("%d", &n);

switch(n)

{

case 1:

system("cls");

outputInformationAboutEmployeesForUser();

system("pause");

break;

case 0:

system("cls");

return;

stop=1;

break;

default:

printf("Wrong number! Try again\n");

break;

}

}

}

void adminPanel() {

int stop=0, n;

while (stop==0)

{

system("cls");

printf("1 - Display information about employees\n2 - Add new employee\n3 - Edit information about employees\n");

printf("4 - Delete employee\n5 - Display all users\n6 - Add new user\n7 - Edit users\n8 - Delete user\n0 - Log out\n");

scanf("%d", &n);

switch(n)

{

case 1:

system("cls");

outputInformationAboutEmployeesForAdmin();

break;

case 2:

system("cls");

addNewEmployee(); break;

case 3:

system("cls");

editInformationAboutEmployees();

break;

case 4:

system("cls");

deleteEmployee();

break;

case 5:

system("cls");

outputUsers();

system("pause");

break;

case 6:

system("cls");

addNewUser();

break;

case 7:

system("cls");

editUsers();

break;

case 8:

system("cls");

deleteUser();

break;

case 0:

system("cls");

return;

stop=1;

break;

default:

system("cls");

printf("\nWrong number! Try again\n");

system("pause");

break;

}

}

}

void outputInformationAboutEmployeesForUser () {

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

printf("%d. | FIO: %s %s %s |", 1+i, a[i].employeeSurname, a[i].employeeName, a[i].employeePatronymic);

printf(" Number of children: %d | Benefit: %.2f\n", a[i].n\_children, a[i].benefit); }

}

void outputInformationAboutEmployeesForAdmin () {

int n, sel;

outputInformationAboutEmployeesForUser();

printf("1 - View information about children\n0-Exit\n");

scanf("%d", &n);

switch(n) {

case 0: break;

case 1: system("cls");

outputInformationAboutEmployeesForUser();

printf("\nSelect number of employee for view information about children (0 - Exit): ");

scanf("%d", &sel);

system("cls");

outputChildren(sel);

system("pause");

break;

default: printf("Wrong number! Try again"); break;

}

}

void addNewEmployee () {

printf("Enter FIO: ");

scanf("%s %s %s", &(a + EmployeeID)->employeeSurname, &(a + EmployeeID)->employeeName, &(a + EmployeeID)->employeePatronymic);

printf("Enter number of children: ");

scanf("%d", &(a + EmployeeID)->n\_children);

system("cls");

if (a[0 + EmployeeID].n\_children == 0)

printf("New Employee was added!");

else {

for(int tmp = 0; tmp < a[0+EmployeeID].n\_children; tmp++) {

printf("Enter FIO of child: ");

scanf("%s %s %s", &b[0 + EmployeeID][0 + tmp].childSurname, &b[0 + EmployeeID][0 + tmp].childName, &b[0 + EmployeeID][0 + tmp].childPatronymic);

printf("\nEnter date of birthday:\nDay: ");

scanf("%d", &b[0 + EmployeeID][0 + tmp].day);

printf("Month: ");

scanf("%d", &b[0 + EmployeeID][0 + tmp].month);

printf("Year: ");

scanf("%d", &b[0 + EmployeeID][0 + tmp].year);

system("cls");

}

printf("New Employee was added!\n");

}

benefitCalculation(EmployeeID);

EmployeeID++;

system("pause");

}

void benefitCalculation(int n) {

int tmp1 = 0;

float tmpbenefit = 0.00;

if (a[0+n].n\_children == 0)

tmpbenefit = 0.00;

else {

for (int tmp = 0; tmp < a[0+n].n\_children; tmp++) {

if ((b[0+n][0+tmp].year == 2019 && b[0+n][0+tmp].month == 12) || (b[0+n][0+tmp].year > 2019)) {

tmp1++;

switch(tmp1) {

case 1: tmpbenefit = tmpbenefit + BASICAMOUNT; break;

case 2: tmpbenefit = tmpbenefit + 1.25\*BASICAMOUNT; break;

default: tmpbenefit = tmpbenefit + 1.5\*BASICAMOUNT; break;

}

}

else

tmpbenefit = tmpbenefit + 0;

}

}

a[0+n].benefit = tmpbenefit;

}

void editInformationAboutEmployees()

{

int stop = 0, n, stop1 = 0, n1 = 0;

while (stop == 0)

{

outputInformationAboutEmployeesForUser();

printf("Enter number of employee for edit(0-Exit): ");

scanf("%d", &n);

if ((n < 0) || (n > EmployeeID))

printf("Wrong number, please try again\n");

else if (n == 0)

stop = 1;

else

while (stop1 == 0)

{ system("cls");

printf("1 - Edit surname\n2 - Edit name\n3 - Edit patronymic\n4 - Edit information about children: \n0 - Exit\n");

scanf("%d", &n1);

switch (n1)

{

case 1:

system("cls");

fflush(stdin);

printf("Enter new surname: ");

scanf("%s", &(a + n - 1)->employeeSurname);

printf("Surname has been changed\n");

system("pause"); break;

case 2:

system("cls");

fflush(stdin);

printf("Enter new name: ");

scanf("%s", &(a + n - 1)->employeeName);

printf("Name has been changed\n");

system("pause"); break;

case 3:

system("cls");

fflush(stdin);

printf("Enter new patronymic: ");

scanf("%s", &(a + n - 1)->employeePatronymic);

printf("Patronymic has been changed\n");

system("pause"); break;

case 4:

editeInformationAboutChildren(n);

case 0:

stop1 = 1; break;

default:

system("cls");

fflush(stdin);

printf("Wrong number, please try again\n");

system("pause"); break;

}

}

}

}

void editeInformationAboutChildren(int n)

{

int n1=0, stop4=0;

system("cls");

fflush(stdin);

while (stop4 == 0) {

outputChildren(n);

printf("1 - Change information about current children\n2 - Add new child\n3 - Delete child\n0 - Exit\n");

scanf("%d", &n1);

switch(n1) {

case 1:

system("cls");

editeInformationAboutCurrentChildren(n); break;

case 2:

system("cls");

addNewChild(n); break;

case 3:

system("cls");

deleteChild (n); break;

case 0:

system("cls");

stop4 = 1; break;

default:

system("cls");

fflush(stdin);

printf("Wrong number, please try again\n");

system("pause");

break;

}

benefitCalculation(n-1); }

}

void outputChildren(int n)

{

for (int i = 0; i < a[n-1].n\_children; i++) {

printf("%d. | FIO: %s %s %s |", i+1, b[n -1][0+i].childSurname, b[n-1][0+i].childName, b[n-1][0+i].childPatronymic);

printf(" Date of birthday: %d.%d.%d\n", b[n-1][0+i].day, b[n-1][0+i].month, b[n-1][0+i].year);

}

}

void editeInformationAboutCurrentChildren(int n) {

int stop3 = 0, sel, n2;

system("cls");

outputChildren(n);

printf("Select number of children: ");

scanf("%d", &sel);

while (stop3 == 0) {

system("cls");

printf("1 - Edit surname\n2 - Edit name\n3 - Edit patronymic\n4 - Edit date of birthday: \n0 - Exit\n");

scanf("%d", &n2);

switch (n2) {

case 1:

system("cls");

fflush(stdin);

printf("Enter new surname: ");

scanf("%s", &b[n-1][sel-1].childSurname);

printf("Surname has been changed\n");

system("pause"); break;

case 2:

system("cls");

fflush(stdin);

printf("Enter new name: ");

scanf("%s", &b[n-1][sel-1].childName);

printf("Name has been changed\n");

system("pause"); break;

case 3:

system("cls");

fflush(stdin);

printf("Enter new patronymic: ");

scanf("%s", &b[n-1][sel-1].childPatronymic);

printf("Patronymic has been changed\n");

system("pause"); break;

case 4:

system("cls");

fflush(stdin);

printf("Day: ");

scanf("%d", &b[n-1][sel-1].day);

printf("Month: ");

scanf("%d", &b[n-1][sel-1].month);

printf("Year: ");

scanf("%d", &b[n-1][sel-1].year);

printf("Day of birthday has been changed\n");

system("pause"); break;

case 0:

system("cls"); stop3 = 1; break;

default:

system("cls");

fflush(stdin);

printf("Wrong number, please try again\n");

system("pause"); break;

}

}

}

void addNewChild(int n)

{

printf("Enter FIO of child: ");

scanf("%s %s %s", &b[n-1][a[n-1].n\_children].childSurname, &b[n-1][a[n-1].n\_children].childName, &b[n-1][a[n-1].n\_children].childPatronymic);

printf("\nEnter date of birthday:\nDay: ");

scanf("%d", &b[n-1][a[n-1].n\_children].day);

printf("Month: ");

scanf("%d", &b[n-1][a[n-1].n\_children].month);

printf("Year: ");

scanf("%d", &b[n-1][a[n-1].n\_children].year);

system("cls");

a[n-1].n\_children++;

}

void deleteChild(int n) {

int stop5 = 0, sel1;

while (stop5 == 0) {

system("cls");

outputChildren(n);

printf("Select number of children(0 - Exit): ");

scanf("%d", &sel1);

if (sel1 == 0)

stop5 = 1;

else

{

for (int i1 = sel1 - 1; i1 < a[n-1].n\_children; i1++)

{

strcpy(b[n-1][i1].childSurname, b[n-1][i1+1].childSurname);

strcpy(b[n-1][i1].childName, b[n-1][i1+1].childName);

strcpy(b[n-1][i1].childPatronymic, b[n-1][i1+1].childPatronymic);

b[n-1][i1].day = b[n-1][i1+1].day;

b[n-1][i1].month = b[n-1][i1+1].month;

b[n-1][i1].year = b[n-1][i1+1].year;

}

a[n-1].n\_children--;

benefitCalculation(n);

printf("Delete child successfully\n");

system("pause");

}

}

}

void deleteEmployee()

{

int stop = 0, n, i;

while (stop == 0)

{

outputInformationAboutEmployeesForUser();

if (EmployeeID > 0)

{

printf("Enter the employee's number(0-Exit): ");

scanf("%d", &n);

if (n == 0)

stop = 1;

else

{

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

{

strcpy((a + i)->employeeSurname, (a + i + 1)->employeeSurname);

strcpy((a + i)->employeeName, (a + i + 1)->employeeName);

strcpy((a + i)->employeePatronymic, (a + i + 1)->employeePatronymic);

(a + i)->n\_children = (a+i+1)->n\_children;

(a + i)->benefit = (a+i+1)->benefit;

for (int k = 0; k < (a+i+1)->n\_children; k++)

b[0+i][0+k] = b[0+i+1][0+k];

}

EmployeeID--;

printf("Delete employee successfully\n");

}

}

else

{

stop = 1;

system("pause");

}

}

}

void outputUsers()

{

system("cls");

fflush(stdin);

printf("A list of users:\n");

for (int i = 0; i < UserID; i++)

printf("%d. Login: %s Password: %s Role: %d\n", i + 1, (a1 + i)->login, (a1 + i)->password, (a1 + i)->role);

}

void addNewUser()

{

int i, stop = 0;

while (stop == 0)

{

system("cls");

fflush(stdin);

printf("Enter login: ");

scanf("%s", &((a1 + UserID)->login));

printf("Enter Password: ");

scanf("%s", &((a1 + UserID)->password));

printf("Enter Role(1-admin or 2-user): ");

scanf("%d", &((a1 + UserID)->role));

if ((a1 + UserID)->role == 1 || (a1 + UserID)->role == 2)

{

printf("New user was added\n");

UserID++;

stop = 1;

system("pause");

}

else

{

printf("Wrong number of role, please try again\n");

system("pause");

}

}

}

void editUsers() {

int stop = 0, n, stop1 = 0, n1, stop2 = 0;

while (stop == 0) {

outputUsers();

printf("Enter number of user for edit(0-Exit): ");

scanf("%d", &n);

if ((n < 0) || (n > UserID))

printf("Wrong number, please try again\n");

else if (n == 0)

stop = 1;

else

while (stop1 == 0) {

system("cls");

printf("1 - Edit login\n2 - Edit password\n3 - Edit role\n0 - Exit\n");

scanf("%d", &n1);

switch (n1)

{

case 1:

system("cls");

printf("Enter new login: ");

scanf("%s", &(a1 + n - 1)->login);

printf("Login has been changed\n");

system("pause"); break;

case 2:

system("cls");

printf("Enter new password: ");

scanf("%s", &(a1 + n - 1)->password);

printf("Password has been changed\n");

system("pause"); break;

case 3:

while (stop2 == 0) {

system("cls");

printf("Enter new role(1-admin or 2-user): ");

scanf("%d", &(a1 + n - 1)->role);

if ((a1 + n - 1)->role == 1 || (a1 + n - 1)->role == 2)

stop2 = 1;

else

printf("Wrong number, please try again\n");

}

printf("Role has been changed\n");

system("pause"); break;

case 0:

stop1 = 1; break;

default:

system("cls");

printf("Wrong number, please try again\n");

system("pause"); break;

}

}

}

}

void deleteUser()

{

int stop = 0, n, i;

while (stop == 0)

{

outputUsers();

if (UserID >= 1)

{

printf("Enter the user's number(0-Exit): ");

scanf("%d", &n);

if (n == 0)

stop = 1;

else

{

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

{

strcpy((a1 + i)->login, (a1 + i + 1)->login);

strcpy((a1 + i)->password, (a1 + i + 1)->password);

(a1 + i)->role = (a1+i+1)->role;

}

UserID--;

printf("Delete user successfully\n");

system("pause");

}

}

else

{

stop = 1;

system("pause");

}

}

}