МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕХНОЛОГИЙ И УПРАВЛЕНИЯ ИМ. К.Г. РАЗУМОВСКОГО (ПЕРВЫЙ КАЗАЧИЙ УНИВЕРСИТЕТ)

(ФГБОУ ВО МГУТУ им. К.Г.РАЗУМОВСКОГО (ПКУ))



Факультет цифровых технологий

Кафедра: Информационные системы и цифровые технологии

Направление подготовки – ­­­­­09.03.01 «Проектирование и разработка программного обеспечения»

ОТЧЕТ ПО ДИСЦИПЛИНЕ

«Основы алгоритмизации и программирования»

Лабораторная работа № 8.

Тема: «Работа со структурами с использованием файлов. Создание меню»

|  |  |
| --- | --- |
| Преподаватель | Таченков О.С. |
| (ученая степень, ученое звание, фамилия, инициалы) |
| Студент | 1 090301-РПРОо-24/1 Асылбек уулу Бакыт |
| курс группа (фамилия, имя, отчество) |

Москва, 2024 г.

**СОДЕРЖАНИЕ**

[СЛОВЕСНАЯ ПОДСТАНОВКА ЗАДАЧИ 3](#_Toc184165306)

[МАТЕМАТИЧЕСКАЯ ПОДСТАНОВКА ЗАДАЧИ 4](#_Toc184165307)

[ЛИСТИНГ ПРОГРАММЫ 5](#_Toc184165308)

[РЕЗУЛЬТАТЫ 9](#_Toc184165309)

# СЛОВЕСНАЯ ПОДСТАНОВКА ЗАДАЧИ

На основе задания и структуры лабораторной работы № 7 доработать программный продукт, добавив следующий функционал:

* сохранение и чтение информации структуры в файл.
* организация меню для работы с файлом следующего вида:

1. Создать файл (записать в него данные)
2. Просмотреть файл (вывести данные из файла на экран)
3. Работа с данными из файла по условию задачи.
4. До записать данные в файл (этот пункт меню – по желанию, не обязательно).
5. Выход.

Указания: вся программа выполняется на основе функций для работы с файлом. Для управления функциями использовать массив указателей на функции. Подход для работы с файлами выбираете самостоятельно. Формат файлов также выбираете самостоятельно. Можно использовать форматы txt, doc, xls и т.д. Для вызова функций меню использовать массив указателей на функцию – обязательное требование!!!

# МАТЕМАТИЧЕСКАЯ ПОДСТАНОВКА ЗАДАЧИ

Дано:

iae.name – имя предприятия,

iae.allEmployees (A) – всего персонала по плану,

iae.industrialEmployees (IE) – кол-во промышленного персонала,

iae.notIndustrialEmployees (NIE) – кол-во непромышленного персонала,

iae.notIndustrialEmployeesPercentage (NIEP) – доля непромышленного персонала.

iae.planForEmployees (P) – план по персоналу.

Отображение нижней границы процента выполнения плана по персоналу:

Отображать записи с процентом выполнения плана по персоналу, большим заданного:

, где p1 заданный от пользователя процент выполнения плана по персоналу.

Поиск предприятия с наименьшей долей непромышленного персонала:

# ЛИСТИНГ ПРОГРАММЫ

struct InformationAboutEnterprise8PW {  
 char\* name;  
 int allEmployees;  
 int industrialEmployees;  
 int notIndustrialEmployees;  
 float notIndustrialEmployeesPercentage;  
 float planForEmployees;  
} typedef infAboutEnterprise8PW;  
infAboutEnterprise8PW\* creatingAndFillingIAEs(int n) {  
 infAboutEnterprise8PW\* enterprises = calloc(n, sizeof(infAboutEnterprise8PW));  
 if (enterprises == **NULL**) {  
 free(enterprises);  
 return nullptr;  
 }  
 for (int i = 0; i < n; i++) {  
 printf("Enterprise %d\n", i + 1);  
 printf("name:");  
 enterprises[i].name = calloc(20, sizeof(char));  
 if (enterprises[i].name == **NULL**) {  
 for (int j = 0; j <= i; j++) free(enterprises[i].name);  
 free(enterprises);  
 return nullptr;  
 }  
 scanf\_s("%s", enterprises[i].name, 20);  
 printf("allEmployees:");  
 scanf\_s("%d", &enterprises[i].allEmployees);  
 printf("industrialEmployees:");  
 scanf\_s("%d", &enterprises[i].industrialEmployees);  
 printf("notIndustrialEmployees:");  
 scanf\_s("%d", &enterprises[i].notIndustrialEmployees);  
 enterprises[i].notIndustrialEmployeesPercentage = ((float)enterprises[i].notIndustrialEmployees / (float)enterprises[i].allEmployees) \* 100;  
 enterprises[i].planForEmployees = ((float)(enterprises[i].industrialEmployees + enterprises[i].notIndustrialEmployees) / (float)enterprises[i].allEmployees) \* 100;  
 }  
 return enterprises;  
}  
infAboutEnterprise8PW\* sortedIAEsByAllEmployees(int n, infAboutEnterprise8PW\* enterprises) {  
 for (int i = 0; i < n - 1; i++) {  
 for (int j = i + 1; j < n; j++) {  
 if (enterprises[i].allEmployees < enterprises[j].allEmployees) {  
 infAboutEnterprise8PW temp = enterprises[i];  
 enterprises[i] = enterprises[j];  
 enterprises[j] = temp;  
 }  
 }  
 }  
 return enterprises;  
}  
void printIAETableHeader() {  
 printf("+-----+-----------+-------+-------+------+---------+-----+\n");  
 printf("| Num | Name | AE | IE | NIE | NIE (%) | PFE |\n");  
 printf("+-----+-----------+-------+-------+------+---------+-----+\n");  
}  
void printIAE(const infAboutEnterprise8PW enterprise, const int i) {  
 printf("| %-3d | %s ", i + 1, enterprise.name);  
 printf("| %d ", enterprise.allEmployees);  
 printf("| %d ", enterprise.industrialEmployees);  
 printf("| %d ", enterprise.notIndustrialEmployees);  
 printf("| %.0f ", enterprise.notIndustrialEmployeesPercentage);  
 printf("| %.0f |\n", enterprise.planForEmployees);  
 printf("+-----+-----------+-------+-------+------+---------+-----+\n");  
}  
char\* smallestNotIndustrialEmployeesPercentageEnterprise(const int n, const infAboutEnterprise8PW\* enterprises) {  
 infAboutEnterprise8PW iae = enterprises[0];  
 int iaeIndex = 0;  
 for (int i = 1; i < n; i++) {  
 if (enterprises[i].notIndustrialEmployeesPercentage < iae.notIndustrialEmployeesPercentage) {  
 iae = enterprises[i];  
 iaeIndex = i;  
 }  
 }  
 printIAE(iae, iaeIndex);  
 return iae.name;  
}  
void printIAEFilteredByPlanForEmployees(int n, const infAboutEnterprise8PW\* enterprises) {  
 float planForEmployees = 0;  
 printf("input planForEmployees (%):");  
 scanf\_s("%f", &planForEmployees);  
 printIAETableHeader();  
 for (int i = 0; i < n; i++) {  
 if (enterprises[i].planForEmployees >= planForEmployees) printIAE(enterprises[i], i);  
 }  
}  
void printLowerBoundOfPlanForEmployees(int n, const infAboutEnterprise8PW\* enterprises) {  
 printIAETableHeader();  
 for (int i = 0; i < n; i++) {  
 if (enterprises[i].planForEmployees < 50.0f) printIAE(enterprises[i], i);  
 }  
}  
void seventhPWTasks(int n, infAboutEnterprise8PW\* enterprises) {  
 if (enterprises == **NULL**) return;  
 int choice = -1;  
 char\* enterpriseName = smallestNotIndustrialEmployeesPercentageEnterprise(n, enterprises);  
 while (choice != 0) {  
 printf("Menu for working with enterprises:\n0 - exit\n1 - lowerBoundOfPlanForEmployees\n2 - printIAEFilteredByPlanForEmployees\n3 - smallestNotIndustrialEmployeesPercentageEnterprise\n4 - print all\n0 - exit\ninput i:");  
 scanf\_s("%d", &choice);  
 switch (choice) {  
 case 1:  
 printLowerBoundOfPlanForEmployees(n, enterprises);  
 break;  
 case 2:  
 printIAEFilteredByPlanForEmployees(n, enterprises);  
 break;  
 case 3:printf("%s\n", enterpriseName);  
 break;  
 case 4:  
 printIAETableHeader();  
 int totalEmployees = 0, totalIndustrialEmployees = 0, totalNotIndustrialEmployees = 0;  
 for (int i = 0; i < n; i++) {  
 printIAE(enterprises[i], i);  
 totalEmployees += enterprises[i].allEmployees;  
 totalIndustrialEmployees += enterprises[i].industrialEmployees;  
 totalNotIndustrialEmployees += enterprises[i].notIndustrialEmployees;  
 }  
 float totalNotIndustrialEmployeesPercentage = ((float)totalNotIndustrialEmployees / (float)totalEmployees) \* 100;  
 float totalPlanForEmployees = ((float)(totalIndustrialEmployees + totalNotIndustrialEmployees) / (float)totalEmployees) \* 100;  
 printf("\n\n");  
 printf("| total:\t|\n");  
 printf("| %d ", totalEmployees);  
 printf("| %d ", totalIndustrialEmployees);  
 printf("| %d ", totalNotIndustrialEmployees);  
 printf("| %.0f ", totalNotIndustrialEmployeesPercentage);  
 printf("| %.0f |\n", totalPlanForEmployees);  
 break;  
 default:  
 break;  
 }  
 }  
 free(enterpriseName);  
}  
int eightPW3Task() {  
 int n = 0;  
 printf("input the number of enterprises:");  
 scanf\_s("%d", &n);  
 if (n <= 0) return printf("error: n <= 0");  
 int choice = -1;  
 FILE\* file = nullptr;  
 infAboutEnterprise8PW\* enterprises8PW, \*enterprises = nullptr;  
 while (choice != 0) {  
 printf("Menu for working with file:\n0 - exit\n1 - create new/(open existing) file and fill enterprises\n2 - print enterprises\n3 - open new menu\ninput i:");  
 scanf\_s("%d", &choice);  
 switch (choice) {  
 case 1:  
 enterprises = creatingAndFillingIAEs(n);  
 if (enterprises == **NULL**) return printf("error: enterprises pointer is NULL");  
 enterprises = sortedIAEsByAllEmployees(n, enterprises);  
 file = fopen("C:\\Users\\fred\\Downloads\\8PW3Task.bin", "wb+");  
 if (file == **NULL**) return printf("error: file not found");  
 for (int i = 0; i < n; i++) fwrite(&enterprises[i], sizeof(infAboutEnterprise8PW), 1, file);  
 break;  
 case 2:  
 enterprises8PW = calloc(n, sizeof(infAboutEnterprise8PW));  
 rewind(file);  
 printIAETableHeader();  
 for (int i = 0; i < n; i++) {  
 fread(&enterprises8PW[i], sizeof(infAboutEnterprise8PW), 1, file);  
 printIAE(enterprises8PW[i], i);  
 }  
 break;  
 case 3:  
 seventhPWTasks(n, enterprises8PW);  
 break;  
 default:  
 printf("Bye Bye!\n");  
 break;  
 }  
 }  
 for (int i = 0; i < n; i++) {  
 if (enterprises[i].name != **NULL**) free(enterprises[i].name);  
 if (enterprises8PW[i].name != **NULL**) free(enterprises8PW[i].name);  
 }  
 if (enterprises != **NULL**) free(enterprises);  
 if (enterprises8PW != **NULL**) free(enterprises8PW);  
 if (file != **NULL**) fclose(file);  
 return 0;  
}

# РЕЗУЛЬТАТЫ











