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 № 7.

Вариант №3

Тема: «Задача со сложной структурой данных»

|  |  |
| --- | --- |
| 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](#_Toc183161905)

[МАТЕМАТИЧЕСКАЯ ПОДСТАНОВКА ЗАДАЧИ 5](#_Toc183161906)

[ЛИСТИНГ ПРОГРАММЫ 6](#_Toc183161907)

[РЕЗУЛЬТАТЫ 9](#_Toc183161908)

# СЛОВЕСНАЯ ПОДСТАНОВКА ЗАДАЧИ

Составить выходную форму со всеми требуемыми выходными данными на основе введенных данных. Для программной организации работы с данными использовать структуры данных. Реализовать в программе:

* создание и заполнение структуры;
* вывод структуры;
* расчет данных по заданию.

По результатам обследования предприятий выяснить процентное выполнение плана по персоналу. Для каждого предприятия известно: наименование предприятия, фактическая численность персонала (промышленного и непромышленного) и плановая численность всего персонала. Число предприятий не более 30. Результаты распечатать в виде таблицы:

Сведения о предприятиях, в которых выполнение плана по персоналу не менее… %

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Наименование предприятия | Всего персонала по плану | Фактически персонала | | Доля непромышленного персонала | Выполнение плана по персоналу |
| промышленного | непромышленного |
| 1 |  |  |  |  |  |  |
| Итого: | |  |  |  |  |  |

Указания: Заполнить исходную таблицу и сортировать ее по уменьшению значений второго столбца. После этого в цикле, пока пользователь не откажется

* запрашивать нижнюю границу процента выполнения плана по персоналу;
* копировать из исходной в рабочую таблицу строки с процентом выполнения плана по персоналу, большим заданного;
* выявлять предприятие с наименьшей долей непромышленного персонала и запоминать его наименование;
* выдавать сведения о предприятиях.

# 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

#include <stdio.h>  
#include <stdlib.h>  
#define n 3  
  
struct InformationAboutEnterprises {  
 char\* name;  
 int allEmployees;  
 int industrialEmployees;  
 int notIndustrialEmployees;  
 float notIndustrialEmployeesPercentage;  
 float planForEmployees;  
} typedef infAboutEnterprises;  
  
infAboutEnterprises\* creatingAndFillingIAEs() {  
 infAboutEnterprises\* enterprises = calloc(n, sizeof(infAboutEnterprises));  
 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;  
 enterprises[i].planForEmployees = (float)(enterprises[i].industrialEmployees + enterprises[i].notIndustrialEmployees) / (float)enterprises[i].allEmployees;  
 }  
 return enterprises;  
}  
infAboutEnterprises\* sortedIAEsByAllEmployees(infAboutEnterprises\* enterprises) {  
 for (int i = 0; i < n - 1; i++) {  
 for (int j = i + 1; j < n; j++) {  
 if (enterprises[i].allEmployees < enterprises[j].allEmployees) {  
 infAboutEnterprises temp = enterprises[i];  
 enterprises[i] = enterprises[j];  
 enterprises[j] = temp;  
 }  
 }  
 }  
 return enterprises;  
}  
void printIAE(const infAboutEnterprises enterprise, const int i) {  
 printf("| %d enterprise", i + 1);  
 printf(" name: %s\t\t|\n", enterprise.name);  
 printf("| allEmployees: %d\t\t\t|\n", enterprise.allEmployees);  
 printf("| industrialEmployees: %d\t\t|\n", enterprise.industrialEmployees);  
 printf("| notIndustrialEmployees: %d\t\t|\n", enterprise.notIndustrialEmployees);  
 printf("| notIndustrialEmployeesPercentage: %.2f|\n", enterprise.notIndustrialEmployeesPercentage);  
 printf("| planForEmployees: %.2f\t\t|\n", enterprise.planForEmployees);  
 printf("-----------------------------------------\n");  
}  
char\* smallestNotIndustrialEmployeesPercentageEnterprise(const infAboutEnterprises\* enterprises) {  
 infAboutEnterprises 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(const infAboutEnterprises\* enterprises) {  
 float planForEmployees = 0;  
 printf("input planForEmployees:");  
 scanf\_s("%f", &planForEmployees);  
 for (int i = 0; i < n; i++) {  
 if (enterprises[i].planForEmployees >= planForEmployees) printIAE(enterprises[i], i);  
 }  
}  
void printLowerBoundOfPlanForEmployees(const infAboutEnterprises\* enterprises) {  
 for (int i = 0; i < n; i++) {  
 if (enterprises[i].planForEmployees < 0.5f) printIAE(enterprises[i], i);  
 }  
}  
int seventhPW3Task() {  
 infAboutEnterprises\* enterprises = creatingAndFillingIAEs();  
 if (enterprises == NULL) return printf("error: enterprises pointer is NULL");  
 enterprises = sortedIAEsByAllEmployees(enterprises);  
 for (int i = 0; i < n; i++) printIAE(enterprises[i], i);  
 printf("\n");  
 int choice = -1;  
 char\* enterpriseName = smallestNotIndustrialEmployeesPercentageEnterprise(enterprises);  
 while (choice != 0) {  
 printf("1 - lowerBoundOfPlanForEmployees\n2 - IAEFilteredByPlanForEmployees\n3 - smallestNotIndustrialEmployeesPercentageEnterprise\n4 - print all\n0 - exit\ninput i:");

scanf\_s("%d", &choice);  
 switch (choice) {  
 case 1:printLowerBoundOfPlanForEmployees(enterprises);  
 break;  
 case 2:printIAEFilteredByPlanForEmployees(enterprises);  
 break;  
 case 3:printf("%s\n", enterpriseName);  
 break;  
 case 4:  
 for (int i = 0; i < n; i++) printIAE(enterprises[i], i);  
 break;  
 default:  
 break;  
 }  
 }  
 free(enterpriseName);  
 for (int i = 0; i < n; i++) free(enterprises[i].name);  
 free(enterprises);  
 return 0;  
}

# РЕЗУЛЬТАТЫ





