МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

Кафедра вычислительной техники

Отчет по лабораторной работе № 9
по дисциплине «Программирование»
Тема: Линейные односвязные списки

Студент гр. 9305 Салауров Е. М.

Преподаватель Перязева Ю. В.

Санкт-Петербург

2020

Оглавление

Введение

Задание

Для выбранной предметной области создать динамический массив структур, содержащих характеристики объектов предметной области.

Обязательный набор полей:

- динамический массив символов, включая пробелы (name)
- произвольный динамический массив символов
- числовые поля типов int и float (не менее двух полей каждого типа)
- поле с числовым массивом.

Написать программу, обеспечивающую начальное формирование массива структур при чтении из файла (текст с разделителями — CSV) с последующим возможным дополнением элементов массива при вводе с клавиатуры. Следует использовать указатели на структуры и указатели на функции обработки массива в соответствии с вариантом задания.

Во всех случаях, когда при поиске записей результат отсутствует, следует вывести сообщение.

Вариант задания номер 8:

Выбор записей, в которых значение любого символьного поля (выбор из меню) начинается с указанной подстроки без учета регистра, сортировка результата по убыванию значений последнего числового поля.

Постановка задачи и описание решения

Для решения этой задачи изначально нужно объявить структуру, я сделал это так:

```
typedef struct ZNAK {
 char NAME[MAXLEN];
 char NIK[MAXLEN];
 int DATE[3];
 int chislopole1;
 int chislopole2;
 float chislopole3;
 float chislopole4;
} group;
```

Далее программа начинает свою работу с проверки на корректное открытие файла для работы с ним. По заданию файл должен быть формата csv, поэтому была объявлена переменная sep, которая является символом разделителем в этом файле.

После успешного открытия файла, программа подсчитывает количество строк в этом файле, и записывает это в переменную number_of_lines. После этого программа выделяет память под структуру, с помошью функции split() программа создает двухмерный массив и выделяет для него память, а с помощью функции struct_fill() заполняет его.

Также программа упорядочивает массив структур по убыванию последнего числового поля, то есть по убыванию переменной chislopole4. Дальше программа просит пользователя выбрать в каком символьном поле он хочет работать, на выбор у пользователя есть два столбца, столбец с именем (NAME) и столбец с никнеймом (NIK). Далее пользователь вводит строчку (bukva), на вхождение которой программа проверяет в одном или другом столбце.

После проведенных операций программа выводит таблицу, которая состоит из элементов массива структур, в символьные поля которых произошло вхождение строчки (bukva). Если же вхождений не произошло программа ничего не выводит.

Описание структуры:

Структура ZNAK:

Название переменной	**Тип переменной**	**Назначение**
NAME	char	Поле с именем
NIK	char	Поле со знаком зоди
DATE	int	Массив содержащий дату рождения
chislopole1	int	Любимое число пользователя
Chislopole2	int	Счастливое число пользователя
Chislopole3	float	Процент удачи пользователя
Chislopole4	float	

Описание функций:

Функция main():

Описание:

Точка входа в программу. Отвечает за открытие файла, содержащего данные

для последующей работы.

Прототип:

int main()

Пример вызова:

main()

Описание переменных:

Название переменной	Тип переменной	Назначение
tab		
ch	group	Массив для чтения данных из файла
slen	int	Переменная для хранения длины массива
i		Переменная для работы оператора фор
number_of_lines		Переменная хранящая количество строчек в файле
s2	char	Массив для заполнения структуры
s1	char	Так же массив для чтения данных из файла
sep	char	Переменная ограничитель для прохода по файлу
df	FILE	Файл
VIVOD	void	Указатель на функцию вывода шапки таблицы

Возвращает значение: 0, если работа программы завершена успешно.

Функция simple_split():

Описание:

Функция разделения строки по заданному разделителю.

Каждая строка файла разделяется на элементы промежуточного массива строк

s2 по разделителям с помощью функции? и в зависимости от типа поля

элемента массива структур выполняется преобразование элемента массива

строк в поле отдельной структуры.

Прототип:

char **simple_split(char *str, int length, char sep)

Пример вызова:

s2=simple_split(s1,slen,sep);

Описание переменных:

Название переменной	Тип переменной	Назначение
str_array	char	Массив в который копируются данные из файла
i	int	Переменная для работы с циклом
j	int	Переменная для работы с циклом
k	int	Переменная для работы с массивом
m	int	Переменная для работы с массивом
key	int	Переменная индикатор
count	int	Переменная для очистки массива

Возвращаемое значение: массив строк.

Функция ClearStringArray():

Описание:

Функция очистки памяти для динамического массива строк.

Прототип:

void ClearStringArray(char **str, int n)

Пример вызова:

ClearStringArray(str_array,count);

Описание переменных:

Название переменной	Тип переменной	Назначение
i	int	Переменная для прохода по массиву
str	char**	Массив строк

Функция struct_fill():

Описание:

Функция заполнения структуры данными из файла. В массив строк вводятся полученные из simple_split данные.

Прототип:

cars *struct_fill(char **str)

Пример вызова:

ch[i]=struct_fill(s2);

Описание переменных:

Название переменной	Тип переменной	Назначение
str0	group	Структура для распределения в ней данных из массива

Контрольные примеры

Пример 1:

Пользователь вводит данные для поиска «vod» и выбирает столбец 2

Пример 2:

Пользователь вводит данные для поиска «RI» и выбирает столбец 2

Пример 3:

Пользователь вводит данные для поиска «MC» и выбирает столбец 1

Пример 4:

Пользователь вводит данные для поиска «ABC» и выбирает столбец 1

Алгоритм на естественном языке

Main

1) проверяет на правильность открытие файла

2) Если файл открыт успешно, то сосчитать количество строк в нём

3) Пока i=0 не дойдет до последней строки:

Вычисляем длину строки из файла

Вызываем функцию для сортировки массива строк по символуразделителю simple_split

Далее программа создает массив и копирует информацию из файла.

4) После этого программа выводит массив в виде таблицы, запрашивает пользователя по какому столбцу ведется поиск данных, какие данные нужно найти и выводит таблицу с элементами, которые удовлетворяют требованию.

simple_split

1) Считаем количество строк

2) Выделяем память для массива строк

Если память выделилась удачно, то

Производится считывание каждой строки, пока не дойдем до

последней

Ищем в каждой строке символ-разделитель и пропускаем его

После очищаем память функцией ClearStringArray

3) Возвращаем полученный массив строк

ClearStringArray

1) Пока i=0 не дойдет до последней строки, освобождаем память каждой

строки из массива строк и присваиваем NULL

struct_fill

1) Выделяем память для массива структур

Если память выделилась удачно, то

Каждой ячейке присваивается значение из массива строк,

полученного ранее

2) Возвращаем структуру

Strcasestr

Функция проходит по строке сравнивая элементы независимо от регистра, если вхождений нет, то возвращает NULL

print_header

1) Вывод шапки структуры

Текст программы

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#define MAXLEN 256

typedef struct ZNAK {
   char NAME[MAXLEN];
   char NIK[MAXLEN];
   int DATE[3];
   int chislopole1;
   int chislopole2;
   float chislopole3;
   float chislopole4;
} group;

// Функция для заполения структуры
char **split(char *str, char sep, int len);

// Очищение двумерного динамического массива
void clear_str_array(char **str, int n);

struct ZNAK struct_fill(struct ZNAK tab, char **s2);

void print_header();

char *strcasestr(const char *str, const char *pattern) {
   size_t i;

   if (!*pattern)
      return (char*)str;
```

```
    for (; *str; str++) {
        if (toupper(*str) == toupper(*pattern)) {
            for (i = 1;; i++) {
                if (!pattern[i])
                    return (char*)str;
                if (toupper(str[i]) != toupper(pattern[i]))
                    break;
            }
        }
    }
    return NULL;
}

int main() {
    group *tab;
    char s1[MAXLEN];
    char **s2;
    int i, check, count;
    int number_of_lines;
    FILE *df;
    char sep = ';';
    if ((df = fopen("struct-data-03.csv", "r")) != NULL) {


        void (*VIVOD)(void);
        VIVOD = print_header;
        number_of_lines = 0;
        while ((fgets(s1, MAXLEN, df)) != NULL) number_of_lines++;
        rewind(df);


        //Выделение памяти под структуру
        if ((tab = malloc(number_of_lines * sizeof(group))) != NULL) {
            //Заполнение структуры
            for (i = 0; i < number_of_lines; i++) {
                fgets(s1, MAXLEN, df);
                s1[strlen(s1)] = '\0';
                s2 = split(s1, sep, strlen(s1));
                tab[i] = struct_fill(tab[i], s2);
            }

            check = 1;
```

```
group tmp;
while (check == 1) {
   check = 0;
   for (i = 0; i < number_of_lines - 1; i++) {
      if (tab[i].chislopole4 > tab[i+1].chislopole4) {
         tmp = tab[i];
         tab[i] = tab[i + 1];
         tab[i + 1] = tmp;
         check = 1;
      }
   }
}
VIVOD();
for (i = 0; i < number_of_lines; i++) {
   printf("|%20s |%10s|%2d|%2d|%2d|%5d|%5d|%10f|%10f|\n",
         tab[i].NAME,
         tab[i].NIK,
         tab[i].DATE[0],
         tab[i].DATE[1],
         tab[i].DATE[2],
         tab[i].chislopole1,
         tab[i].chislopole2,
         tab[i].chislopole3,
         tab[i].chislopole4);
}

char bukva[125];
int stolbec;

printf("\nDannie dla poiska: ");
gets(bukva);

count = 0;
printf("Kakoi stolbec vibrat` (1- NAME, 2- NIC): ");
scanf("%d", &stolbec);

for (i = 0; i < number_of_lines; i++) {

   if ((stolbec == 1) && (strcasestr(tab[i].NAME, bukva)) != NULL) {
      count++;
      if (count == 1) VIVOD();
```

```
                    printf("|%20s  |%10s|%2d|%2d|%2d|%5d|%5d|%10f|%10f|\n", tab[i].NAME, tab[i].NIK, tab[i].DATE[0], tab[i].DATE[1], tab[i].DATE[2], tab[i].chislopole1, tab[i].chislopole2, tab[i].chislopole3, tab[i].chislopole4);
                }
                if ((stolbec == 2) && (strcasestr(tab[i].NIK, bukva)) != NULL) {
                    count++;
                    if (count == 1) VIVOD();
                    printf("|%20s  |%10s|%2d|%2d|%2d|%5d|%5d|%10f|%10f|\n", tab[i].NAME, tab[i].NIK, tab[i].DATE[0], tab[i].DATE[1], tab[i].DATE[2], tab[i].chislopole1, tab[i].chislopole2, tab[i].chislopole3, tab[i].chislopole4);
                }
            }
            clear_str_array(s2, 9);
            free(tab);
            tab = NULL;
        } else {
            free(tab);
            tab = NULL;
        }
    } else puts("Unable to open file!");
    return 0;
}

struct ZNAK struct_fill(struct ZNAK tab, char **s2) {
    strcpy(tab.NAME, s2[0]);
    strcpy(tab.NIK, s2[1]);
    tab.DATE[0] = atoi(s2[2]);
    tab.DATE[1] = atoi(s2[3]);
    tab.DATE[2] = atoi(s2[4]);
    tab.chislopole1 = atoi(s2[5]);
    tab.chislopole2 = atoi(s2[6]);
    tab.chislopole3 = atof(s2[7]);
    tab.chislopole4 = atof(s2[8]);
    return tab;
}

char **split(char *str, char sep, int len) {
    int i, m, n;
    int check, count, number_of_words;
    char **p;
    // Подсчет слов в строке
    number_of_words = 0;
```

```
    for (i = 0; i < len; i++) {
        if (str[i] == sep) number_of_words++;
    }
    check = 0;
    // Выделение памяти для двумерного массива p
    if ((p = malloc((number_of_words + 1) * sizeof(char *))) != NULL) {
        for (i = 0, count = 0; i <= number_of_words; i++, count++) {
            if ((p[i] = malloc(len * sizeof(char))) != NULL) {
                check = 1;
            } else {
                check = 0;
                i = number_of_words;
            }
        }
        if (check) {
            n = 0;
            m = 0;
            for (i = 0; i < len; i++) {
                if (str[i] != sep) p[n][i - m] = str[i];
                else {
                    p[n][i - m] = '\0';
                    m = i + 1;
                    n++;
                }
            }
        } else
        {
            puts("ERROR at string allocation!\n");
            clear_str_array(p, count);;
        }
    }
    return p;
}

void clear_str_array(char **str, int n) {
    int i;
    for (i = 0; i < n; i++) {
        free(str[i]);
        str[i] = NULL;
    }
    free(str);
    str = NULL;
```

```
}

void print_header() {
   printf("|  %20s|%10s|%10s|%5s|%5s|%10s|%10s|\n",  "NAME",  "Zod",  "Date",  "int1",
"int2", "float1", "float2");
   printf("+--------------------+----------+----+--+--+-----+-----+----------+----------+\n");
}
```

Пример работы программы

```
|                NAME|       Zod|      Date| int1| int2|    float1|    float2|
+--------------------+----------+----+--+--+-----+-----+----------+----------+
|         Mae Vanessa |       lev|1990| 2| 1|  123|  123|  0.220000|  0.124000|
|       McCartney Pol |       lev|1990| 2| 1|   23|  342|  0.220000|  0.124000|
|       McCartney Pol |       lev|1990| 2| 1|  454|   45|  0.220000|  0.124000|
|         Mae Vanessa |       lev|1990| 2| 1|   23|   45|  0.220000|  0.124000|
|         Mae Vanessa |  ckorpion|1990| 2| 1|   13| 4454|  0.220000|  0.124000|
|         Mae Vanessa |   vodolei|1990| 2| 1|   45|   45|  0.220000|  0.124000|
|      Howston Witney |   vodolei|1987| 6| 1|    1|    3|  0.660000|  0.164000|
|      Howston Witney |   vodolei|1987| 6| 1|    1|   23|  0.660000|  0.164000|
|        Chang Jeckie |      oven|1987| 6| 1|   45| 4454|  0.660000|  0.164000|
|          Last James |   vodolei|2000| 3| 2|    1|   12|  0.120000|  0.234000|
|         Mae Vanessa |   vodolei|2000| 3| 2|   45|  245|  0.330000|  0.234000|
|          Last James |  ckorpion|2000| 3| 2| 4454|  454|  0.330000|  0.234000|
|       McCartney Pol |  ckorpion|1999| 8| 2|    3|    1|  0.880000|  0.284000|
|       McCartney Pol |  ckorpion|1999| 8| 2|    3|   13|  0.880000|  0.284000|
|       McCartney Pol |      oven|1999| 8| 2| 4454|  454|  0.880000|  0.284000|
|         Mae Vanessa |       lev|1999| 8| 2|  454|   23|  0.880000|  0.284000|
|       McCartney Pol |   vodolei|1999| 8| 2|   23|   45|  0.880000|  0.284000|
|      Stone Sharonne |      riba|2001| 5| 3|  234|    3|  0.550000|  0.354000|
|      Stone Sharonne |      riba|2001| 5| 3|   13|    2|  0.550000|  0.354000|
|      Stone Sharonne |      riba|2001| 5| 3|   23|    1|  0.550000|  0.354000|
|      Stone Sharonne |      riba|2001| 5| 3|   13| 4454|  0.550000|  0.354000|
|          Last James |      riba|2001| 5| 3|   45|   45|  0.550000|  0.354000|
|      Stone Sharonne |      riba|2001| 5| 3|   23|   45|  0.550000|  0.354000|
|      Stone Sharonne |       lev|2001| 5| 3|  454|   23|  0.550000|  0.354000|
|        Chang Jeckie |      oven|1999| 1| 4|  213|   34|  0.110000|  0.414000|
|          Last James |      oven|1999| 1| 4|   45|    3|  0.110000|  0.414000|
|        Chang Jeckie |      riba|1999| 1| 4|  454|   23|  0.110000|  0.414000|
|      Stone Sharonne |  ckorpion|1999| 8| 6| 4454|   43|  0.880000|  0.684000|

Dannie dla poiska: VOD
Kakoi stolbec vibrat` (1- NAME, 2- NIC): 2
|                NAME|       Zod|      Date| int1| int2|    float1|    float2|
+--------------------+----------+----+--+--+-----+-----+----------+----------+
|         Mae Vanessa |   vodolei|1990| 2| 1|   45|   45|  0.220000|  0.124000|
|      Howston Witney |   vodolei|1987| 6| 1|    1|    3|  0.660000|  0.164000|
|      Howston Witney |   vodolei|1987| 6| 1|    1|   23|  0.660000|  0.164000|
|          Last James |   vodolei|2000| 3| 2|    1|   12|  0.120000|  0.234000|
|         Mae Vanessa |   vodolei|2000| 3| 2|   45|  245|  0.330000|  0.234000|
|       McCartney Pol |   vodolei|1999| 8| 2|   23|   45|  0.880000|  0.284000|

Process returned 0 (0x0)   execution time : 6.707 s
Press any key to continue.
```

Заключение

Выводы:

При выполнении лабораторной работы были получены практические навыки в разработке алгоритма и написании программы на языке Си. Были получены основные знания о синтаксисе языка Си, в частности, о программировании задач со структурами, а также правилах написания кода на языке Си.