**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра математического обеспечения и применения ЭВМ**

**КУРСОВАЯ РАБОТА**

**по дисциплине «Программирование»**

Тема: Обработка текстовых файлов в формате .csv

|  |  |  |
| --- | --- | --- |
| Студент гр. 6304 |  | Зыль С. Е. |
| Преподаватель |  | Берленко Т.А. |

Санкт-Петербург

2017

**ЗАДАНИЕ**

**на курсовую работу**

|  |  |  |
| --- | --- | --- |
| Студент Зыль С. Е. | | |
| Группа 6304 | | |
| Тема работы: Обработка текстовых файлов в формате .csv | | |
|  | | |
| Содержание пояснительной записки:   * Содержание * Введение * Описание функций, необходимых для работы программы * Примеры работы программы * Заключение * Список использованных источников * Приложение А. Исходный код программы | | |
| Объем пояснительной записки: | | |
| Дата выдачи задания: 28.04.2017 | | |
| Дата сдачи: | | |
| Дата защиты: | | |
| Студент |  | Зыль С. Е. |
| Преподаватель |  | Берленко Т.А. |

**Аннотация**

В данной работе была создана программа на языке программирования C, которая производит обработку csv-файлов посредством набора функций. Были созданы и описаны необходимые функции, позволяющие считать строки, содержащиеся в файлах, проводить поиск определенных данных в файлах, сохранять данные из файлов. Также были приведены примеры работы программы и полное описание исходного кода.

**Оглавление**

[Введение 2](#_Toc484013737)

[Цель работы 2](#_Toc484013738)

[Формулировка задачи 2](#_Toc484013739)

[Решение задачи 3](#_Toc484013740)

[1.1. Создание структуры данных 3](#_Toc484013741)

[1.2. Функция инициализации элемента списка 3](#_Toc484013742)

[1.3. Функция чтения первой таблицы 4](#_Toc484013743)

[1.4. Функция чтения второй таблицы 4](#_Toc484013744)

[1.5. Функция нахождения максимальной оценки 5](#_Toc484013745)

[1.6. Функция поиска количество плохоуспевающих студентов 5](#_Toc484013746)

[1.7. Функция main 6](#_Toc484013747)

[1.8. Makefile 6](#_Toc484013748)

[Работа с репозиторием 7](#_Toc484013749)

[Примеры работы программы 7](#_Toc484013750)

[Пример работы программы при корректных входных данных 7](#_Toc484013751)

[Пример работы программы при отсутствии одного из входных файлов 8](#_Toc484013752)

[Заключение 9](#_Toc484013753)

[Список использованных источников 10](#_Toc484013754)

[Приложение А 11](#_Toc484013755)

# Введение

Цель работы

Создание проекта для обработки данных csv файлов. Изучение методов работы с csv файлами.

## Формулировка задачи

Для достижения поставленной цели требуется решить следующую задачу:

Требуется написать программу, на вход которой подаются две таблицы хрянящиеся в csv файлах. Первая таблица содержит информацию о ФИО студента и его Github аккаунте, email'е и номере группы. Вторая таблица содержит фамилию, имя, количество баллов за экзамен. Программа должна находить количество студентов, которые написали хуже, чем 60% от максимального балла и выводить результат на консоль.

Программа получает параметры из входного потока. Параметры:

* input\_file\_1 - csv файл
* input\_file\_2 - csv файл

В случае, если программа получила некорректные параметры, то:

* выводится сообщение об ошибке “Fail with <имя параметра>”.

# Решение задачи

## 1.1. Создание структуры данных

С помощью типа *struct* объявим новый сложный тип *Student*, для упорядочения генерируемых данных.

struct Student {

char last\_name[20];

char first\_name[20];

char fathers\_name[20];

char github[20];

char mail[20];

int group;

int result;

struct Student\* next;

};

typedef struct Student Student;

## . Функция инициализации элемента списка

В данную функцию передается указатель на строку таблицы. Далее динамически выделяем память по элемент списка, массив слов, который будет получен при помощи функции strtok. Функция strtok разделяет строку на лексемы, в данном случае разделителями являются “,” и “;”. После того как массив array\_of\_words заполнился, происходит инициализация списка. Функция возвращает указатель на структуру Student.

Student\* createStudent(char\* array) { // функция инициализации элемента списка

Student\* new\_Student = (Student\*)malloc(sizeof(Student));

char\* array\_of\_words[50];

char\* ptr = strtok(array, "; ,"); // разделение строки на лексемы

int i = 0;

while (ptr) {

array\_of\_words[i++] = ptr;// заполнение массива лексемами

ptr = strtok(NULL, "; ,");

}

strcpy(new\_Student->last\_name, array\_of\_words[0]);

strcpy(new\_Student->first\_name, array\_of\_words[1]);

strcpy(new\_Student->fathers\_name, array\_of\_words[2]);

strcpy(new\_Student->github, array\_of\_words[3]);

strcpy(new\_Student->mail, array\_of\_words[4]);

new\_Student->group = atoi(array\_of\_words[5]); // atoi преобразует char\* в int

return new\_Student;

}

## 1.3. Функция чтения первой таблицы

В функцию передается файл, с которого считываются строки, которые заполняют массив array\_of\_strings. Из файла c помощью функции fgets считывается строки до “\n”. Эти строки помещаются в массив указателей array\_of\_strings. Одновременно происходит создание однонаправленного списка с помощью функции createStudent, в которую поочередно передаются элементы array\_of\_strings. В конце происходит освобождение памяти. Функция возвращает указатель на первый элемент списка.

Student\* readingTable\_1(FILE\* table) {

char array[500];

char\* array\_of\_strings[50]; int i = 0;

Student\* current;

Student\* prev;

while (fgets(array, 500, table)) {

if (array[strlen(array) - 1] == '\n')

array[strlen(array) - 1] = '\0';

array\_of\_strings[i] = array;

if (i == 0) {

current = createStudent(\*array\_of\_strings);

current->next = NULL;

prev = current;

}

else {

current = createStudent(array\_of\_strings[i]);

current->next = prev;

prev = current;

}

i++;

}

fclose(table);

return current;

}

## Функция чтения второй таблицы

В функцию передается файл, из которого будут считываться оценки студентов, и указатель на первый элемент созданного списка. Аналогично функции для первой таблицы мы считываем данные в уже созданный список. Так как столбец оценок идет третьим в таблице, то мы инициализируем поле структуры result, только кода счетчик key = 2. При помощи функции atoi идет преобразование оценки к типу int. Функция ничего не возвращает.

void readingTable\_2(FILE\* table, Student\* head) {

char array[50];

char\* array\_of\_strings[50];

int i = 0;

Student\* current;

current = head;

while(current){

fgets(array, 500, table);

if (array[strlen(array) - 1] == '\n')

array[strlen(array) - 1] = '\0';

array\_of\_strings[i] = array;

array\_of\_strings[i] = strtok(array\_of\_strings[i], ";,");

int key = 0;

while (array\_of\_strings[i]) {

if (key<2) {

array\_of\_strings[i] = strtok(NULL, ";,");

}

else {

current->result = atoi(array\_of\_strings[i]);

array\_of\_strings[i] = strtok(NULL, ";,");

}

key++;

}

## Функция нахождения максимальной оценки

Функция получает указатель на первый элемент списка. В цикле происходит поиск максимального элемента, который будет присвоен переменной max. Балл студента записывает в переменную result. Если result > max, то происходит присваивание: max = result. Так функция проходит весь список и возвращает max.

int maxGrade(Student\* head) {

int result = 0;

int max = 0;

while (head) {

result = head->result;

if (result>max)

max = result;

head = head->next;

}

return max;

}

## Функция поиска количество плохоуспевающих студентов

Функция получает указатель на первый элемент списка и выполняет подсчет количества студентов, оценка которых ниже, чем 60% максимального результата, который был получен путем вызова функции maxGrade. Функция проходится по списку и увеличивает счетчик counter, если оценка данного студента меньше переменной max. Функция возвращает max.

int badGrades(Student\* head) {

int counter = 0;

int max = maxGrade(head)\*0.6;

while (head) {

if (head->result < max) {

counter++;

}

head = head->next;

}

return counter;

}

## Функция main

В функции main подключен русский алфавит функцией setlocale(0, “”). Открываются две таблицы. Если одна из таблиц не была получена, выводится сообщение об ошибке, и программа прекращает работу. Далее происходит создание списка с помощью вызова функций readingTable\_1 и readingTable\_2. После формирование списка, происходит вызов функции badGrades, значение которой выводится на экран.

int main() {

setlocale(0, "");

FILE\* table1 = fopen(INPUT\_FILE\_1, "r");

FILE\* table2 = fopen(INPUT\_FILE\_2, "r");

if (table1 == NULL) {

printf("Fail with INPUT\_FILE\_1\n");

return 0;

}

else if (table2 == NULL) {

printf("Fail with INPUT\_FILE\_2\n");

return 0;

}

Student\* head = readingTable\_1(table1);

readingTable\_2(table2, head);

int counter = badGrades(head);

printf("The number of students whose result is worse than 60%% of the maximum score: %d\n", counter);

system("pause");

fclose(table1);

fclose(table2);

return 0;

}

## Makefile

Создадим поочерёдно объектные файлы функций и главной функции, после скомпилируем их вместе, также предусмотрена функция очистки после сборки проекта.

#Главная цель

curs\_work: main.o Functions.o

gcc main.o - o curs\_work Functions.o

rm \*.o

#Объектный файл для функции main

main.o : main.c

gcc - c main.c

#Объектный файл для функций

Functions.o : Functions.c Functions.h

gcc - c Functions.c

# Работа с репозиторием

1. Скопируем репозиторий группы 6304

*Git clone https://github.com/moevm/pr1-2016-6304.git*

1. От ветки master создаем локальную ветку с названием работы и именем:

*Git checkout –b “Zyl\_kurswork\_sem2”*

1. В папку с клонированным репозиторием скопируем файлы с работой с помощью команды *cp –r*
2. Индексируем добавленные файлы командой

*git add \*.c \*.h makefile*

1. Коммитим наши изменения, чтобы подготовить их к отправке на сервер

*Git commit –m “Zyl kurswork done”*

1. Отправляем изменения в репозиторий командой

*Git push origin Zyl\_kurswork\_sem2*

1. На сайте github создаем *pull request* в ветку *master.*

# Примеры работы программы

## Пример работы программы при корректных входных данных

|  |  |  |
| --- | --- | --- |
| input\_file\_1 | input\_file\_2 | Результат |
| Ageev; Fedor;Ivanovich;fedya;ran@gmail.com;1123  Lobov;Igor;Artemovich;beeL;ll@mail.ru;2234  Runin;Mike;Victorovich;LOAL;peofk@mail.ru;3254  Ivanov;Ivan;Ivanovich;guke;pop\_@gmail.com;1111  Rotov;Artem;Sergeevich;root;pixar@mail.ru;2834  Ionova;Kate;Petrovna;artcone;bb@mail.ru;1324  Morkina;Anna;Igorevna;lqlq;cps@gmail.com;2538  Folina;Nastya;Vitalevna;lalala;rand@gmail.com;4328  Petrova;Vika;Andreevna;girl;GQ@mail.ru;1532  Novikova;Vitalina ;Ivanovna;brandon;lili@gmail.com;2435 | Ageev; Fedor;30  Lobov;Igor;20  Runin;Mike;100  Ivanov;Ivan;90  Rotov;Artem;55  Ionova;Kate;22  Morkina;Anna;60  Folina;Nastya;71  Petrova;Vika;19  Novikova;Vitalina ;55 | 6 |
| Степанова,Полина,Созоновна,Полина@github.ru,Степанова@mail.ru,6303  Мясникова,Екатерина,Федосеевна,Екатерина@github.ru,Мясникова@mail.ru,6303  Потапов,Вячеслав,Павлович,Вячеслав@github.ru,Потапова@mail.ru,6303  Герасимов,Игнатий,Парфеньевна,Игнатий@github.ru,Герасимова@mail.ru,6304  Кузьмин,Василий,Протасьевна,Василий@github.ru,Кузьмина@mail.ru,6304  Попов,Юлиан,Романович,Юлиан@github.ru,Попова@mail.ru,6304  Мартынова,Надежда,Егоровна,Надежда@github.ru,Мартынов@mail.ru,6304  Третьякова,Элеонора,Денисович,Элеонора@github.ru,Третьяков@mail.ru,6304 | Попов,Юлиан,32  Степанова,Полина,78  Третьякова,Элеонора,71  Мартынова,Надежда,78  Мясникова,Екатерина,78  Потапов,Вячеслав,32  Герасимов,Игнатий,20  Кузьмин,Василий,12 | 4 |

## Пример работы программы при отсутствии одного из входных файлов

|  |  |  |
| --- | --- | --- |
| input\_file\_1 | input\_file\_2 | Результат |
| //Отсутствует | Ageev; Fedor;30  Lobov;Igor;20  Runin;Mike;100  Ivanov;Ivan;90  Rotov;Artem;55  Ionova;Kate;22  Morkina;Anna;60  Folina;Nastya;71  Petrova;Vika;19  Novikova;Vitalina ;55 | **Fail with INPUT\_FILE\_1** |
| Ageev; Fedor;Ivanovich;fedya;ran@gmail.com;1123  Lobov;Igor;Artemovich;beeL;ll@mail.ru;2234  Runin;Mike;Victorovich;LOAL;peofk@mail.ru;3254  Ivanov;Ivan;Ivanovich;guke;pop\_@gmail.com;1111  Rotov;Artem;Sergeevich;root;pixar@mail.ru;2834  Ionova;Kate;Petrovna;artcone;bb@mail.ru;1324  Morkina;Anna;Igorevna;lqlq;cps@gmail.com;2538  Folina;Nastya;Vitalevna;lalala;rand@gmail.com;4328  Petrova;Vika;Andreevna;girl;GQ@mail.ru;1532 | //Отсутствует | **Fail with INPUT\_FILE\_2** |

# Заключение

В ходе работы была создана программа на языке программирования С для обработки двух csv-файлов. Чтобы выполнить поставленную задачу, были созданы и описаны структура и все необходимые функции для считывания таблиц и поиска определенных данных. Помимо этого, были приведены примеры работы программы и полное описание исходного кода.

# Список использованных источников

1. Язык программирования СИ / Керниган Б., Ритчи Д. СПб.: Издательство "Невский Диалект", 2001. 352 с.

2. UNIX. Программное окружение / Керниган Б., Пайк Р. СПб.: Символ Плюс, 2003. 416 с.

# Приложение А

**Исходный код программы**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <math.h>

#include <locale.h>

#define INPUT\_FILE\_1 "t1.csv"

#define INPUT\_FILE\_2 "t2.csv"

struct Student { // структура, описывающая информацию в csv файлах

char last\_name[20];

char first\_name[20];

char fathers\_name[20];

char github[20];

char mail[20];

int group;

int result;

struct Student\* next;

};

typedef struct Student Student; // спецификатор "typedef" делает новое имя типа

Student\* createStudent(char\* array) { // функция инициализации элемента списка

Student\* new\_Student = (Student\*)malloc(sizeof(Student));

char\* array\_of\_words[50];

char\* ptr = strtok(array, "; ,"); // разделение строки на лексемы

int i = 0;

while (ptr) {

array\_of\_words[i++] = ptr;// заполнение массива лексемами

ptr = strtok(NULL, "; ,");

}

strcpy(new\_Student->last\_name, array\_of\_words[0]);

strcpy(new\_Student->first\_name, array\_of\_words[1]);

strcpy(new\_Student->fathers\_name, array\_of\_words[2]);

strcpy(new\_Student->github, array\_of\_words[3]);

strcpy(new\_Student->mail, array\_of\_words[4]);

new\_Student->group = atoi(array\_of\_words[5]); // atoi преобразует char\* в int

return new\_Student;

}

/\*Функция чтения первой таблицы\*/

Student\* readingTable\_1(FILE\* table) {

char array[500];

char\* array\_of\_strings[50];

int i = 0;

Student\* current;

Student\* prev;

while (fgets(array, 500, table)) {

if (array[strlen(array) - 1] == '\n')

array[strlen(array) - 1] = '\0';

array\_of\_strings[i] = array;

if (i == 0) {

current = createStudent(\*array\_of\_strings);

current->next = NULL;

prev = current;

}

else {

current = createStudent(array\_of\_strings[i]);

current->next = prev;

prev = current;

}

i++;

}

fclose(table);

return current;

}

/\*Функция чтения второй таблицы\*/

void readingTable\_2(FILE\* table, Student\* head) {

char array[500];

char\* array\_of\_strings[50];

int i = 0;

Student\* current;

current = head;

while(current){

fgets(array, 500, table);

if (array[strlen(array) - 1] == '\n')

array[strlen(array) - 1] = '\0';

array\_of\_strings[i] = array;

array\_of\_strings[i] = strtok(array\_of\_strings[i], ";,");

int key = 0;

while (array\_of\_strings[i]) {

if (key<2) {

array\_of\_strings[i] = strtok(NULL, ";,");

}

else {

current->result = atoi(array\_of\_strings[i]);

array\_of\_strings[i] = strtok(NULL, ";,");

}

key++;

}

current = current->next;

i++;

}

fclose(table);

}

/\*Функция нахождения максимальной оценки среди студентов\*/

int maxGrade(Student\* head) {

int result = 0;

int max = 0;

while (head) {

result = head->result;

if (result>max)

max = result;

head = head->next;

}

return max;

}

/\*Функция нахождения количества студентов, оценка которых ниже, чем 60% от максимальной оценки\*/

int badGrades(Student\* head) {

int counter = 0;

int max = maxGrade(head)\*0.6;

while (head) {

if (head->result < max) {

counter++;

}

head = head->next;

}

return counter;

}

int main() {

setlocale(0, "");

FILE\* table1 = fopen(INPUT\_FILE\_1, "r");

FILE\* table2 = fopen(INPUT\_FILE\_2, "r");

if (table1 == NULL) {

printf("Fail with INPUT\_FILE\_1\n");

return 0;

}

else if (table2 == NULL) {

printf("Fail with INPUT\_FILE\_2\n");

return 0;

}

Student\* head = readingTable\_1(table1);

readingTable\_2(table2, head);

int counter = badGrades(head);

printf("The number of students whose result is worse than 60%% of the maximum score: %d\n", counter);

fclose(table1);

fclose(table2);

return 0;

}