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

Кафедра вычислительной техники

Отчет по лабораторной работе № 13
по дисциплине «Программирование»
Тема: Битовые поля в структурах.

Студентка гр. 9305 Ковалева К.В.

Преподаватель Перязева Ю.В.

Санкт-Петербург

2020

Содержание

Введение

Цель работы – получить практические навыки в разработке алгоритма и написании программы на языке Си для знакомства с синтаксисом, в частности, с работой со структурами, содержащими битовые поля, а также правилами написания кода на языке Си.

Задание

Написать программу учета сдачи зачетов при помощи битовых полей. Структура содержит поля: фамилия, группа, зачеты (битовое поле, 4 бита). Предусмотреть ввод списка из файла и с клавиатуры, определить, сколько зачетов сдано.

Постановка задачи и описание решения

Дана структура с именем BITS (список студентов, которые прошли 4 зачетных единицы), содержащая поля: Name – фамилия студента, Group – номер группы, Passed credits – четыре зачета (1 – сдал, 0 - нет), Result – результат («сдал», если пройден минимальный порог, вводимый пользователем, «должник» в ином случае). Пользователю представляется выбор: загрузить данные из CSV-файла или, если такового нет, ввести собственноручно, с клавиатуры.

В случае удачного открытия файла в этом файле подсчитывается количество строк, после чего выделяется нужный объем памяти для массива структур. При успешном выделении памяти строки из файла считываются по одной в переменную s1, каждая строка файла разделяется на элементы массива строк s2 по разделителям с помощью функции simple_split, а затем структура заполняется полученными данными. После заполнения полей элемента массива структур промежуточный массив строк очищается с помощью функции clear_str_array. Ввод данных с клавиатуры осуществляется за счет функции input_line, заполнение структуры - с помощью функции str_fill. Далее пользователь видит исходный список студентов, а затем ему

предлагается ввести минимальное количество сданных зачетов. Структура проходит через цикл: если битовое значение ненулевое, счетчик увеличивается на единицу. Если итоговое значение счетчика меньше введенного лимита, студент – должник, иначе – сдал. Вердикт записывается в столбец «Result» и список студентов выводится. Под результирующим списком выведено число сдавших и не сдавших.

Программа завершает свою работу, освобождая память, выделенную для массива структур, с использованием функции free.

Описание структуры

Таблица 1. Описание пользовательских типов данных CARS - структуры, описывающей информационное поле элемента списка.

Поле	Тип	Назначение
name	char*	Фамилия студента
group	char*	Тип кузова
a0, a1, a2, a3	unsigned	Зачеты
result	char*	Результат

Описание функций

1. Функция main

Описание:

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

Прототип:

int main()

Пример вызова:

main()

Описание переменных:

Имя	Тип	Назначение
st1	studs*	Строка, содержащая структурные данные
slen	int	Длина всей строки
i	int	Параметр цикла
n	int	Количество строк
limit	int	Минимальное количество сданных зачетов
count	int	Счетчик количества сданных зачетов
flag	int	Флаг, определяющий наличие файла
dolg	int	Количество должников
pass	int	Количество сдавших
s1	char	Строка
s2	char**	Массив строк
sep	char	Символ-разделитель
df	FILE*	Указатель на файл для последующего чтения

Возвращает значение: 0, если работа программы завершена успешно.

2. Функция choose

Описание:
Функция, возвращающая выбор пользователя
Прототип:
int choose();
Пример вызова:
flag=choose();
Описание переменных:

Имя	Тип	Назначение
line	char*	Адрес строки, введенной пользователем
ans_coose	int	Числовой ввод пользователя

Возвращает значение: числового ввода пользователя.

3. Функция input_line

Описание:
Функция, осуществляющая считывание данных с клавиатуры.
Прототип:
char* input_line();
Пример вызова:
str->name=input_line();
Описание переменных:

Имя	Тип	Назначение
line_input	char*	Адрес первого элемента вводимой строки

Возвращает значение: line_input.

4. Функция str_fill

Описание:
Функция, заполняющая структуру при вводе данных с клавиатуры.
Прототип:
cars* str_fill();
Пример вызова:
sled->content=str_fill();
Описание переменных:

Имя	Тип	Назначение
str	bits*	Адрес структуры, в котором содержится информация об автомобиле

s	char*	Вспомогательная переменная
cop	int	Переменная для хранения ввода пользователя

Возвращает значение: str.

5. Функция clear_str_array

Описание:

Функция, очищающая структуры.

Прототип:

void clear_str_array(char **str, int n);

Пример вызова:

clear_str_array(s2,7);

Описание переменных:

Имя	Тип	Назначение
i	int	Параметр цикла

6. Функция simple_split

Описание:

Функция, делящая строку по знаку 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 **split(char *data0, char sep);

Пример вызова:

str=split(data, sep);

Описание переменных:

Имя	Тип	Назначение
str	char*	Строка, содержащая структурные данные
i	int	Параметр цикла
j	int	Параметр цикла
k	int	Параметр для столбца
m	int	Количество символов разделителей в строке
key	int	Флаг для выделения памяти
count	int	Количество строк

Возвращаемое значение: массив строк.

Контрольные примеры

Входные данные: 2

Jam;1000;1;1;1;1;-;
Jam;1000;1;1;1;0;-;
Jam;1000;1;1;0;0;-;
Jam;1000;1;0;0;0;-;
Jam;1000;0;0;0;0;-;

Выходные данные:

Jam;1000;1;1;1;1;сдал;
Jam;1000;1;1;1;0;сдал;
Jam;1000;1;1;0;0;сдал;
Jam;1000;1;0;0;0;должник;
Jam;1000;0;0;0;0;должник;
В данном списке 3 сдавших и 2 должников.

Входные данные: 0

Jam;1000;1;1;1;1;-;
Jam;1000;1;1;1;0;-;
Jam;1000;1;1;0;0;-;
Jam;1000;1;0;0;0;-;
Jam;1000;0;0;0;0;-;

Выходные данные:

Jam;1000;1;1;1;1;сдал;
Jam;1000;1;1;1;0;сдал;
Jam;1000;1;1;0;0;сдал;
Jam;1000;1;0;0;0;сдал;
Jam;1000;0;0;0;0;сдал;
В данном списке 5 сдавших и 0 должников.

Входные данные: 5

Jam;1000;1;1;1;1;-;
Jam;1000;1;1;1;0;-;
Jam;1000;1;1;0;0;-;
Jam;1000;1;0;0;0;-;
Jam;1000;0;0;0;0;-;

Выходные данные:

Jam;1000;1;1;1;1;должник;
Jam;1000;1;1;1;0;должник;
Jam;1000;1;1;0;0;должник;
Jam;1000;1;0;0;0;должник;
Jam;1000;0;0;0;0;должник;
В данном списке 0 сдавших и 5 должников.

Алгоритм на естественном языке

Начало.
Шаг 1. Считывание данных из CSV-файла. Если такого файла не существует, производится ввод данных с клавиатуры.
Шаг 2. Вывод исходного списка студентов.
Шаг 3. Ввод пользователем минимального порога сдачи зачетов.
Шаг 4. Обработка, подсчитывание зачетов каждого студента, запись итога в «результат».
Шаг 7. Вывод результирующего списка, завершение работы программы.
Конец.

Текст программы

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <locale.h>


/// Определение структурного типа

typedef struct b

{

    char* name;             ///Фамилия студента

    char* group;            ///Номер группы

    unsigned a0: 1;         ///Зачет

    unsigned a1: 1;         ///Зачет

    unsigned a2: 1;         ///Зачет

    unsigned a3: 1;         ///Зачет

    char* result;           ///Результат

} bits;
```

```
///Функция, возвращающая выбор пользователя
int choose();

///Функция, осуществлющая считывание
char* input_line();

///Функция, заполняющая структуру с клавиатуры
bits* str_fill();

/// Функция очистки памяти для динамического массива строк
void clear_str_array(char **str, int n);

/// Функция разделения строки по заданному разделителю
char **simple_split(char *str, int length, char sep);

int main()
{
    bits **st1;                ///Строка, содержащая структурные
данные
    int flag,              ///Флаг, определяющий наличие файла
        slen,              ///Длина всей строки
        i,                 ///Параметр цикла
        n,                 ///Количество строк
```

```
        limit,                  ///Минимальное количество сданных
зачетов
        count,              ///Счетчик количества сданных зачетов
        dolg,               ///Количество должников
        pass;               ///Количество сдавших
    char **s2=NULL;         ///Массив строк
    char s1[128];           ///Срока
    char sep;               ///Символ-разделитель
    FILE *df;               ///Указатель на файл

    setlocale(LC_ALL,"RUS");
    n=0;
    flag=0;
    dolg = 0;
    pass = 0;
    df=fopen("data.csv","r");
    if(df!=NULL)
    {
        /// Подсчет строк в файле
        while(fgets(s1,128,df)!= NULL) n++;
        rewind(df); // Возврат вт
        st1=(bits**)malloc(n*sizeof(bits*));
        if(st1)
        {
            ///Чтение данных из файла и заполнение полей структур
```

```
            sep=';';
            for(i=0,count=0;i<n;i++, count++)
            {
                st1[i]=(bits*)malloc(sizeof(bits));
                if (st1[i])
                {
                    fgets(s1,128,df);
                    slen=strlen(s1);
                    s1[slen-1]='\0';
                    slen=strlen(s1);
                    s2=simple_split(s1,slen,sep);
                    st1[i]-
>name=(char*)malloc((strlen(s2[0])+1)*sizeof(char));
                    st1[i]-
>group=(char*)malloc((strlen(s2[1])+1)*sizeof(char));
                    st1[i]-
>result=(char*)malloc((strlen(s2[6])+1)*sizeof(char));
                    if(s2&&st1[i]->name&&st1[i]->group&&st1[i]-
>result)
                    {
                        strcpy(st1[i]->name,s2[0]+'\0');
                        strcpy(st1[i]->group,s2[1]+'\0');
                        st1[i]->a0=atoi(s2[2]);
                        st1[i]->a1=atoi(s2[3]);
                        st1[i]->a2=atoi(s2[4]);
                        st1[i]->a3=atoi(s2[5]);
```

```
                strcpy(st1[i]->result,s2[6]+'\0');
                clear_str_array(s2,7);
            }
            else
            {
                i=n;
                puts("Row data not available!");
                system("pause");
                system("cls");
                flag=1;
            }
        }
        else
        {
            puts("Error of memory allocation");
            system("pause");
            system("cls");
            i=n;
            flag=1;
        }
    }
}
else
{
```

```
            puts("Out of memory!");
            system("pause");
            system("cls");
        }
        fclose(df);
    }
    else
    {
        perror("Data error!");
        system("pause");
        system("cls");
        st1=(bits**)malloc(n*sizeof(bits*));
        if(st1)
        {
            i=-1;
            do
            {
                i++;
                puts("Введите данные с клавиатуры:");
                st1[i]=str_fill();
                printf("Хотите продолжить ввод? (1)");
                flag=choose();
            }while(flag==1);
        }
```

```
    }

    if (flag)
    {
        for(i=0;i<=count;i++)
        {
            free(st1[i]->group);
            free(st1[i]->name);
            free(st1[i]->result);
            free(st1[i]);
        }
        free(st1);
    }

    /// Обработка массива структур
    if(st1&&n)
    {
        puts("Список студентов:");
        printf("|%20s |%6s | %22s | %8s|\n",
        "NAME","GROUP","PASSED CREDITS", "RESULT");
         printf("+--------------------+------+----------------
--------+---------+\n");
        for(i=0;i<n;i++)
        {
            printf("|%20s |%6s |%5d %5d %5d %5d | %7s |\n",
```

```
            st1[i]->name, st1[i]->group, st1[i]->a0, st1[i]->a1,
st1[i]->a2, st1[i]->a3, st1[i]->result);
        }
    }
    else
        puts("No data found!");
    system("pause");
    system("cls");

    printf("Введите минимальный порог сданных зачетов - ");
    scanf("%d", &limit);

    printf("|%20s |%6s | %22s | %8s|\n",
    "NAME","GROUP","PASSED CREDITS", "RESULT");
    printf("+--------------------+-------+--------------------
---+---------+\n");
    for(i=0;i<n;i++)
    {
        count=0;
        if(st1[i]->a0) count++;
        if(st1[i]->a1) count++;
        if(st1[i]->a2) count++;
        if(st1[i]->a3) count++;

        if(count < limit)
```

```
        {
            strcpy(st1[i]->result, "должник");
            dolg++;
        }
        else
        {
            strcpy(st1[i]->result, "сдал");
            pass++;
        }
        printf("|%20s |%6s |%5d %5d %5d %5d | %7s |\n",
        st1[i]->name,  st1[i]->group,  st1[i]->a0,  st1[i]->a1,
st1[i]->a2, st1[i]->a3, st1[i]->result);
    }
    printf("В данном списке %d сдавших и %d должников.", pass,
dolg);
    return 0;
}


char **simple_split(char *str, int length, char sep)
{
    char **str_array=NULL;
    int i,j,k,m;
    int key,count;
    for(j=0,m=0;j<length;j++)
    {
```

```
    if(str[j]==sep) m++;
}

key=0;
str_array=(char**)malloc((m+1)*sizeof(char*));
if(str_array!=NULL)
{
    for(i=0,count=0;i<=m;i++,count++)
    {
        str_array[i]=(char*)malloc(length*sizeof(char));
        if(str_array[i]!=NULL) key=1;
        else
        {
            key=0;
            i=m;
        }
    }
    if(key)
    {
        k=0;
        m=0;
        for(j=0;j<length;j++)
        {
            if(str[j]!=sep) str_array[m][j-k]=str[j];
```

```
                else
                {
                    str_array[m][j-k]='\0';
                    k=j+1;
                    m++;
                }
            }
            str_array[m][j-k]='\0';
        }
        else
        {
            clear_str_array(str_array,count);
        }
     }
     return str_array;
}

int choose()
{
    char* line;           ///Адрес строки, введенной пользователем
    int ans;                ///Числовой ввод пользователя

    line=(char*)malloc(15*sizeof(char));
    if (line!=NULL)
```

```
    {
        fflush(stdin);
        fgets(line, 15, stdin);
        ans=atoi(line);
    }
    else
    {
        puts("Ошибка выделения памяти");
        ans=5;
    }

    free(line);

    return ans;
}

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;
}


char* input_line()
{
    char* line_input;           ///Адрес первого элемента вводимой
строки


    line_input=(char*)malloc(80*sizeof(char));
    if (line_input!=NULL)
    {
        fflush(stdin);
        fgets(line_input, 79, stdin);
        line_input[strlen(line_input)-1]='\0';
        line_input=(char*)realloc(line_input,
(strlen(line_input)+1)*sizeof(char));
    }
    else
    {
        puts("Ошибка выделения памяти");
        system("pause");
        system("cls");
    }
    return line_input;
```

```
}

bits* str_fill()
{
    bits *b;            ///Адрес структуры, в которой содержится
информация о мотоцикле

    b=(bits*)malloc(sizeof(bits));
    if (b)
    {
        printf("Введите фамилию студента - ");
        b->name=input_line();

        printf("Введите номер группы - ");
        b->group=input_line();

        printf("Введите результат сдачи первого зачета - ");
        b->a0=choose();

        printf("Введите результат сдачи второго зачета - ");
        b->a1=choose();

        printf("Введите результат сдачи третьего зачета - ");
        b->a2=choose();
```

```
        printf("Введите результат сдачи четвертого зачета - ");
        b->a3=choose();

        printf("В этой графе будет выведен результат сдачи зачетов. Пожалуйства, введите какой-нибудь символ.");
        b->result=input_line();
    }
    else
    {
        puts("Ошибка выделения памяти");
        system("pause");
        system("cls");
    }
    return b;
}
```

Пример работы программы

Исходные данные:

Jam	1000	1	1	1	1	-
Jam	1000	1	1	1	0	-
Jam	1000	1	1	0	0	-
Jam	1000	1	0	0	0	-
Jam	1000	0	0	0	0	-

Вывод программы: Контрольный пример 1

```
Введите минимальный порог сданных зачетов - 2
|                 NAME | GROUP |          PASSED CREDITS |   RESULT|
+--------------------+-------+-------------------------+---------+
|                  Jam |  1000 |    1     1     1     1 |    сдал |
|                  Jam |  1000 |    1     1     1     0 |    сдал |
|                  Jam |  1000 |    1     1     0     0 |    сдал |
|                  Jam |  1000 |    1     0     0     0 | должник |
|                  Jam |  1000 |    0     0     0     0 | должник |
В данном списке 3 сдавших и 2 должников.
Process returned 0 (0x0)   execution time : 4.193 s
Press any key to continue.
```

Контрольный пример 2

```
Введите минимальный порог сданных зачетов - 0
|                 NAME | GROUP |          PASSED CREDITS |   RESULT|
+--------------------+-------+-------------------------+---------+
|                  Jam |  1000 |    1     1     1     1 |    сдал |
|                  Jam |  1000 |    1     1     1     0 |    сдал |
|                  Jam |  1000 |    1     1     0     0 |    сдал |
|                  Jam |  1000 |    1     0     0     0 |    сдал |
|                  Jam |  1000 |    0     0     0     0 |    сдал |
В данном списке 5 сдавших и 0 должников.
Process returned 0 (0x0)   execution time : 3.859 s
Press any key to continue.
```

Заключение

При выполнении лабораторной работы были получены практические навыки в разработке алгоритма и написании программы на языке Си, в частности, в работе со структурами и битовыми полями, чтением данных из CSV-файла и клавиатуры, а также в работе с функциями в структурах.