Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт

Кафедра «Программная инженерия»
кафедра

ОТЧЕТ О ПРАКТИЧЕСКОЙ РАБОТЕ № 5

Рекурсия
Тема

Руководитель _____ А.С. Черниговский
Подпись, дата Инициалы, Фамилия

Студент КИ19-17/1Б, №031939174 _____ А.К. Никитин
Номер группы, зачетной книжки Подпись, дата Инициалы, Фамилия

Красноярск 2020

1 Цель

Ознакомиться с принципом построения и функционирования структур, использованием динамических массивов и файлов для хранения структур написать программу по варианту.

2 Задачи

Для выполнения лабораторной работы необходимо выполнить следующие задачи:

1) выполнить работу в соответствии с заданием;

2) предусмотреть следующие операции: добавление, удаление и вывод;

3) реализовать меню пользователя;

4) экземпляры структур хранить в динамическом массиве;

5) разбить программу на функции (использование глобальных переменных не приветствуется);

6) добавить возможность хранения информации в файле;

7) добавить сортировку по выбранному пользователем полю;

8) не допускается использование глобальных переменных;

9) добавить фильтрацию и поиск по одному или нескольким полям на выбор пользователя.

3 Описание задания

Реализовать структуру, описывающую дату/время. Создать следующие поля: год, месяц, день, час, минута, секунда, стандарт. Добавить вычисляемые поля: преобразование в стандарт UTC, вычисления промежутка между двумя датами, преобразование к строке.

4 Ход выполнения

Ниже представлен листинг программы по заданию.

Листинг 1 – Операции с датой и временем

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <locale.h>
#include <string.h>
#include <time.h>

struct filterOutput
{
    int length;
    struct tm* dateArray;
    int error;
};

// Ввод целого числа
void inputNat(int* number)
{
    while (!scanf("%d", number))
    {
        fflush(stdin);
        printf("Введите корректные данные!\n");
    }
    if (*number < 0)
    {
        printf("Число должно быть натуральным или 0!\n");
        inputNat(number);
    }
}

// Ввод строки произвольной
void inputString(char** word)
{
```

Продолжение листинга 1

```
    int count = 0;
    char inputChar = 0;

    fflush(stdin);

    while(1)
    {
        inputChar = getchar();
        if (inputChar == '\n')
            break;
        else
        {
            *word = realloc(*word, count + 1);
            (*word)[count] = inputChar;
            count++;
        }
    }
    (*word)[count] = '\0';
}

// Проверяет правильность ввода стандарта кодирования
char* checkStandard(char* standard)
{
    int timezoneUTC = 0;

    standard = strlwr(standard);
    // Стандарт  UNIX
    if (strcmp(standard, "unix") == 0)
    {
        return "unix";
    }
        // Стандарт UTC. Если начинаеся со строки "utc"
    else if (strstr(standard, "utc") != NULL && strcmp(strstr(standard, "utc"),
standard) == 0)
    {
        // Мировое время
        if (strlen(standard) == 3)
            return "utc";

            // Часовой пояс
```

Продолжение листинга 1

```
        else if (strlen(standard) == 6 && (standard[3] == '+' || standard[3] ==
'-'))
        {
            timezoneUTC = atoi(standard + 3);
            if (timezoneUTC == 0 || timezoneUTC > 12 || timezoneUTC < -12)
            {
                printf("Вы ввели неправильный часовой пояс (правильные от -12 до
12)!\n");
                return NULL;
            }
            return standard;
        }
    }
    printf("Неверный формат стандарта!\n");
    return NULL;
}

// Преобразовывает локальное время к мировому
struct tm UTC2UTC(struct tm date, char* standard)
{
    int tempDate = 0;
    int timezoneUTC = 0;

    // Если время не мировое, берется поправка на временной пояс
    if (strlen(standard) > 3)
        timezoneUTC = atoi(standard + 3);

    // Преобразование
    tempDate = mktime(&date);
    tempDate -= timezoneUTC * 3600 + timezone; // Глобальная переменная timezone
из библиотеки time.h хранит в себе
    date = *gmtime(&tempDate);  // разницу локального времени компьютера с мировым

    return date;
}

// Преобразовывает время в секундах (стандарт UNIX) в мировое
struct tm UNIX2UTC(int seconds)
{
    struct tm date;
```

Продолжение листинга 1

```
    date = *gmtime(&seconds);

    return date;
}

// Удаляет из массива дату по порядковому номеру
struct tm* deleteDate(struct tm* dateBase, int length, int index)
{
    struct tm* newDateBase;
    newDateBase = (struct tm*) malloc(sizeof(struct tm) * (length-1));

    // Создание нового массива без учета удаляемого элемента
    for (int i = 0; i < length; i++)
    {
        if (i >= index)
            newDateBase[i] = dateBase[i+1];
        else
            newDateBase[i] = dateBase[i];
    }

    return newDateBase;
}

// Функция сортирует массив дат по определенному ключу по возрастанию или убыванию
// В функции будет много однотипного кода для каждого ключа, избавление от которого
усложнило бы код
struct tm* sortArray(struct tm* dateBase, int length, char* key, int order)
{
    struct tm temp;

    if (strcmp(key, "seconds") == 0)
    {
        // Сортировка пузырьком
        for (int i = 0; i < length; i++)
            for (int j = i; j < length; j++)
                if (order)  // По возрастанию
                {
                    if (dateBase[i].tm_sec > dateBase[j].tm_sec)
                    {
                        temp = dateBase[i];
                        dateBase[i] = dateBase[j];
```

Продолжение листинга 1

```
                    dateBase[j] = temp;
                }
            }
            else  // По убыванию
            if (dateBase[i].tm_sec < dateBase[j].tm_sec)
            {
                temp = dateBase[i];
                dateBase[i] = dateBase[j];
                dateBase[j] = temp;
            }
    }

    else if (strcmp(key, "minutes") == 0)
    {
        for (int i = 0; i < length; i++)
            for (int j = i; j < length; j++)
                if (order)  // По возрастанию
                {
                    if (dateBase[i].tm_min > dateBase[j].tm_min)
                    {
                        temp = dateBase[i];
                        dateBase[i] = dateBase[j];
                        dateBase[j] = temp;
                    }
                }
                else  // По убыванию
                if (dateBase[i].tm_min < dateBase[j].tm_min)
                {
                    temp = dateBase[i];
                    dateBase[i] = dateBase[j];
                    dateBase[j] = temp;
                }
    }

    else if (strcmp(key, "hours") == 0)
    {
        for (int i = 0; i < length; i++)
            for (int j = i; j < length; j++)
                if (order)  // По возрастанию
                {
                    if (dateBase[i].tm_hour > dateBase[j].tm_hour)
```

Продолжение листинга 1

```
                {
                    temp = dateBase[i];
                    dateBase[i] = dateBase[j];
                    dateBase[j] = temp;
                }
            }
            else  // По убыванию
            if (dateBase[i].tm_hour < dateBase[j].tm_hour)
            {
                temp = dateBase[i];
                dateBase[i] = dateBase[j];
                dateBase[j] = temp;
            }
}

else if (strcmp(key, "monthday") == 0)
{
    for (int i = 0; i < length; i++)
        for (int j = i; j < length; j++)
            if (order)  // По возрастанию
            {
                if (dateBase[i].tm_mday > dateBase[j].tm_mday)
                {
                    temp = dateBase[i];
                    dateBase[i] = dateBase[j];
                    dateBase[j] = temp;
                }
            }
            else  // По убыванию
            if (dateBase[i].tm_mday < dateBase[j].tm_mday)
            {
                temp = dateBase[i];
                dateBase[i] = dateBase[j];
                dateBase[j] = temp;
            }
}

else if (strcmp(key, "weekday") == 0)
{
    for (int i = 0; i < length; i++)
        for (int j = i; j < length; j++)
```

Продолжение листинга 1

```
            if (order)  // По возрастанию
            {
                if (dateBase[i].tm_wday > dateBase[j].tm_wday)
                {
                    temp = dateBase[i];
                    dateBase[i] = dateBase[j];
                    dateBase[j] = temp;
                }
            }
            else  // По убыванию
            if (dateBase[i].tm_wday < dateBase[j].tm_wday)
            {
                temp = dateBase[i];
                dateBase[i] = dateBase[j];
                dateBase[j] = temp;
            }
    }

    else if (strcmp(key, "month") == 0)
    {
        for (int i = 0; i < length; i++)
            for (int j = i; j < length; j++)
            if (order)  // По возрастанию
            {
                if (dateBase[i].tm_mon > dateBase[j].tm_mon)
                {
                    temp = dateBase[i];
                    dateBase[i] = dateBase[j];
                    dateBase[j] = temp;
                }
            }
            else  // По убыванию
            if (dateBase[i].tm_mon < dateBase[j].tm_mon)
            {
                temp = dateBase[i];
                dateBase[i] = dateBase[j];
                dateBase[j] = temp;
            }
    }

    else if (strcmp(key, "year") == 0)
```

Продолжение листинга 1

```
    {
        for (int i = 0; i < length; i++)
            for (int j = i; j < length; j++)
                if (order)  // По возрастанию
                {
                    if (dateBase[i].tm_year > dateBase[j].tm_year)
                    {
                        temp = dateBase[i];
                        dateBase[i] = dateBase[j];
                        dateBase[j] = temp;
                    }
                }
                else  // По убыванию
        if (dateBase[i].tm_year < dateBase[j].tm_year)
        {
            temp = dateBase[i];
            dateBase[i] = dateBase[j];
            dateBase[j] = temp;
        }
    }

    else
        return NULL;

    return dateBase;
}
// Основной алгоритм для фильтрации, который будет использоваться в функции ниже
struct filterOutput filterAlgorithm(struct tm* dateBase, int length, char* key)
{
    int number = 0;
    int keyLen = 0;
    int count = 0;
    int filterVar = 0;
    struct tm* newDateBase = NULL;

    struct filterOutput output;

    for (int j = 0; j < length; j++)
    {
        // В переменной key хранится поле для фильтра. Данный условный оператор
адаптирует алгоритм под каждое из полей
```

Продолжение листинга 1

```
        if (strstr(key, "seconds") != NULL)
        {
            filterVar = dateBase[j].tm_sec;
            keyLen = strlen("seconds");
        }
        else if (strstr(key, "minutes") != NULL)
        {
            filterVar = dateBase[j].tm_min;
            keyLen = strlen("minutes");
        }
        else if (strstr(key, "hours") != NULL)
        {
            filterVar = dateBase[j].tm_hour;
            keyLen = strlen("hours");
        }
        else if (strstr(key, "weekday") != NULL)
        {
            filterVar = dateBase[j].tm_wday + 1;
            keyLen = strlen("weekday");
        }
        else if (strstr(key, "monthday") != NULL)
        {
            filterVar = dateBase[j].tm_mday;
            keyLen = strlen("monthday");
        }
        else if (strstr(key, "month") != NULL)
        {
            filterVar = dateBase[j].tm_mon + 1;
            keyLen = strlen("month");
        }
        else if (strstr(key, "year") != NULL)
        {
            filterVar = dateBase[j].tm_year + 1900; // Времяисчисление начинается
с 1900 года
            keyLen = strlen("year");
        }

        number = atoi(key + keyLen + 1);

        if (number == 0)
        {
```

Продолжение листинга 1

```
            output.length = 0;
            output.dateArray = NULL;
            output.error = 1;
            return output;
        }

        if (key[keyLen] == '>')
        {
            if (filterVar > number)
            {
                newDateBase = realloc(newDateBase, sizeof(struct tm) * (count +
1));
                newDateBase[count] = dateBase[j];
                count++;
            }
        }

        if (key[keyLen] == '<')
        {
            if (filterVar < number)
            {
                newDateBase = realloc(newDateBase, sizeof(struct tm) * (count +
1));
                newDateBase[count] = dateBase[j];
                count++;
            }
        }

        if (key[keyLen] == '=')
        {
            if (filterVar == number)
            {
                newDateBase = realloc(newDateBase, sizeof(struct tm) * (count +
1));
                newDateBase[count] = dateBase[j];
                count++;
            }
        }
    }

    output.length = count;
```

Продолжение листинга 1

```
    output.dateArray = newDateBase;
    output.error = 0;
    return output;
}

// Главная функция фильтрации
struct filterOutput filter(struct tm* dateBase, int length, char* key)
{
    int count = 0;
    int keysNumber = 0;
    int keyLen = 0;
    int checker = 0;
    char *correctKey = NULL;
    char **keys = NULL;

    struct filterOutput output;

    // Если ключ состоит только из пробелов
    if (strspn(key, " ") == strlen(key))
    {
        output.dateArray = NULL;
        output.length = 0;
        output.error = 1;
        return output;
    }

    // Удаление пробелов из строчки
    for (int i = 0; i < strlen(key); i++)
        if (key[i] != ' ')
        {
            count++;
            correctKey = realloc(correctKey, sizeof(char) * (count));
            correctKey[count - 1] = key[i];
        }
    correctKey[count] = '\0';
    // Разделение строки по запятым
    keys = (char **) malloc(sizeof(char *));
    keys[0] = strtok(correctKey, ",");

    while (keys[keysNumber] != NULL)
    {
```

Продолжение листинга 1

```
        keysNumber++;
        keys = realloc(keys, sizeof(char *) * (keysNumber + 1));
        keys[keysNumber] = strtok(NULL, ",");
    }

    // Цикл для каждого из условий
    for (int i = 0; i < keysNumber; i++)
    {
        keyLen = strlen("seconds");
        // Если начинается с seconds, следующие после него символы - >, < или =,
и после них еще что-то должно быть
        if (strstr(keys[i], "seconds") != 0 &&
            (keys[i][keyLen] == '=' || keys[i][keyLen] == '>' || keys[i][keyLen]
== '<')
            && strlen(keys[i]) > keyLen)
        {
            output = filterAlgorithm(dateBase, length, keys[i]);
            length = output.length;
            memcpy(dateBase, output.dateArray, sizeof(struct tm) * length);
            checker = 1;
        }

        keyLen = strlen("minutes");
        if (strstr(keys[i], "minutes") != 0 &&
            (keys[i][keyLen] == '=' || keys[i][keyLen] == '>' || keys[i][keyLen]
== '<')
            && strlen(keys[i]) > keyLen)
        {
            output = filterAlgorithm(dateBase, length, keys[i]);
            if (output.error != 0)
            {
                length = output.length;
                memcpy(dateBase, output.dateArray, sizeof(struct tm) * length);
                checker = 1;
            }
        }

        keyLen = strlen("hours");
        if (strstr(keys[i], "hours") != 0 &&
            (keys[i][keyLen] == '=' || keys[i][keyLen] == '>' || keys[i][keyLen]
== '<')
```

Продолжение листинга 1

```
            && strlen(keys[i]) > keyLen)
        {
            output = filterAlgorithm(dateBase, length, keys[i]);
            if (output.error != 0)
            {
                length = output.length;
                memcpy(dateBase, output.dateArray, sizeof(struct tm) * length);
                checker = 1;
            }
        }

        keyLen = strlen("weekday");
        if (strstr(keys[i], "weekday") != 0 &&
            (keys[i][keyLen] == '=' || keys[i][keyLen] == '>' || keys[i][keyLen]
== '<')
            && strlen(keys[i]) > keyLen)
        {
            output = filterAlgorithm(dateBase, length, keys[i]);
            if (output.error != 0)
            {
                length = output.length;
                memcpy(dateBase, output.dateArray, sizeof(struct tm) * length);
                checker = 1;
            }
        }

        keyLen = strlen("monthday");
        if (strstr(keys[i], "monthday") != 0 &&
            (keys[i][keyLen] == '=' || keys[i][keyLen] == '>' || keys[i][keyLen]
== '<')
            && strlen(keys[i]) > keyLen)
        {
            output = filterAlgorithm(dateBase, length, keys[i]);
            if (output.error != 0)
            {
                length = output.length;
                memcpy(dateBase, output.dateArray, sizeof(struct tm) * length);
                checker = 1;
            }
        }
```

Продолжение листинга 1

```
        keyLen = strlen("month");
        if (strstr(keys[i], "month") != 0 &&
            (keys[i][keyLen] == '=' || keys[i][keyLen] == '>' || keys[i][keyLen]
== '<')
            && strlen(keys[i]) > keyLen)
        {
            output = filterAlgorithm(dateBase, length, keys[i]);
            if (output.error != 0)
            {
                length = output.length;
                memcpy(dateBase, output.dateArray, sizeof(struct tm) * length);
                checker = 1;
            }
        }

        keyLen = strlen("year");
         if (strstr(keys[i], "year") != 0 && (keys[i][keyLen] == '=' ||
keys[i][keyLen] == '>' || keys[i][keyLen] == '<')
            && strlen(keys[i]) > keyLen)
        {
            output = filterAlgorithm(dateBase, length, keys[i]);
            if (output.error != 0)
            {
                length = output.length;
                memcpy(dateBase, output.dateArray, sizeof(struct tm) * length);
                checker = 1;
            }
        }
    }
    if (checker == 0)
    {
        output.length = 0;
        output.dateArray = NULL;
        output.error = 1;
    }
    return output;
}

int main()
{
```

Продолжение листинга 1

```
    int userChoice = 0;
    int objectsInArray = 0;
    long int second = 0;

    struct tm currentDate;

    struct tm* dateBase = NULL;

    char* standard = NULL;

    setlocale(LC_ALL, "");

    enum Case {dateInput = 1, transform2UTC, calculateInterval, chooseDate,
dataDel, dataOutput,  sort, filterArray,
        loadData, dataSave, exitProg};

    FILE *file;

    do
    {
        printf("\nРАБОТА С ДАТОЙ:                                    РАБОТА С МАССИВОМ
ДАТ:                        РАБОТА С ФАЙЛОМ:\n"
               "1. Ввести дату.                                  4. Выбрать дату из
массива.              9. Загрузить дату из файла.\n"
               "2. Преобразовать в стандарт UTC.             5. Удалить дату из
массива.              10. Сохранить дату в файл из массива.\n"
               "3. Вычислить промежуток.                      6. Вывести массив
дат.\nмежду двумя датами.                "
               "       7. Отсортировать даты в массиве.\n"
               "                                             8. Отфильтровать даты в
массиве.\n\n\n"
               "                                        11. ВЫЙТИ ИЗ ПРОГРАММЫ\n");
        inputNat(&userChoice);

        if (userChoice < 1 || userChoice > 11)
        {
            printf("Введите значение от 1 до 11!\n");
            continue;
        }

        switch (userChoice)
```

Продолжение листинга 1

```
        {
            // Ввод даты с клавиатуры. Дата автоматически добавится в массив с
датами
            case(dateInput):
                {
                    char* correctStandard = NULL;
                    standard = NULL;

                    printf("Введите стандарт времени: UNIX, UTC или UTCxxx (н.п.
UTC+07)\n");
                    inputString(&standard);

                    correctStandard = checkStandard(standard);
                    if (correctStandard == NULL)
                        break;

                    // Ввод даты стандарта UNIX, т.е. время представлено в виде
секунд, начиная с 1970 года
                    if (strcmp(correctStandard, "unix") == 0)
                    {
                        printf("Введите количество секунд:\n");
                        inputNat(&second);

                        dateBase = realloc(dateBase, sizeof(struct tm) *
(objectsInArray + 1));
                        dateBase[objectsInArray] = UNIX2UTC(second);
                        objectsInArray++;

                        printf("Дата успешно создана.\n");
                        break;
                    }

                        // Ввод мирового времени или локального времени. Условие:
если начинается с "utc"
                    else if (strcmp(strstr(standard, "utc"), standard) == 0)
                    {
                        printf("Введите номер дня:\n");
                        inputNat(&currentDate.tm_mday);
                        printf("Введите порядковый номер дня недели:\n");
                        inputNat(&currentDate.tm_wday);
                        currentDate.tm_wday--;
```

Продолжение листинга 1

```
                        printf("Введите месяц:\n");
                        inputNat(&currentDate.tm_mon);
                        currentDate.tm_mon--;
                        printf("Введите год:\n");
                        inputNat(&currentDate.tm_year);
                        currentDate.tm_year -= 1900;
                        printf("Введите количество часов:\n");
                        inputNat(&currentDate.tm_hour);
                        printf("Введите количество минут:\n");
                        inputNat(&currentDate.tm_min);
                        printf("Введите количество секунд:\n");
                        inputNat(&currentDate.tm_sec);

                        // Если asctime не удается перевести дату в строку, он
возвращает NULL
                        if (!asctime(&currentDate))
                        {
                            printf("Введенная вами информация неверная!\n");
                            break;
                        }

                        dateBase = realloc(dateBase, sizeof(struct tm) *
(objectsInArray + 1));
                        dateBase[objectsInArray] = UTC2UTC(currentDate,
standard); // Данные в массиве в UTC
                        objectsInArray++;
                        break;
                    }
                }

                // Преобразует текущую дату в формат UTC
            case (transform2UTC):
            {
                char* dateString = NULL;

                if (standard == NULL)
                {
                    printf("Сначала введите дату!\n");
                    break;
                }
```

Продолжение листинга 1

```
            // UTC
        else if (strcmp(standard, "utc") == 0)
        {
            printf("Введенная дата уже в формате UTC!\n");
            break;
        }

            // UNIX
        else if (strcmp(standard, "unix") == 0)
        {
            currentDate = UNIX2UTC(second);
            strcpy(standard, "utc");

            // Вывод даты в двух форматах
            printf("UNIX: %d\n", second);
            dateString = asctime(&currentDate);
            printf("UTC: %s\n", dateString);
            break;
        }

            // UTCxxx
        else if (strlen(standard) == 6 && (standard[3] == '+' ||
standard[3] == '-'))
        {
            // Строчка до преобразования
            dateString = asctime(&currentDate);
            printf("%s: %s\n", strupr(standard), dateString);

            currentDate = UTC2UTC(currentDate, standard);
            strcpy(standard, "utc");

            //Строчка после преобразования
            dateString = asctime(&currentDate);
            printf("UTC: %s\n", dateString);

            break;
        }
        else
        {
            printf("С введенным стандартом что-то не так...");
```

Продолжение листинга 1

```
                    break;
                }
            }

                // Изначально по заданию требовалось посчитать промежуток между двумя датами, но я решил, что промежуток
                // между введенной датой и настоящим временем будет интереснее и в той же степени отражает задание
            case(calculateInterval):
            {
                struct tm date;

                time_t currentTime = time(NULL);  // Текущее время
                time_t interval;

                int enteredTime = 0;

                if (standard == NULL)
                {
                    printf("Сначала введите дату!\n");
                    break;
                }

                // Для вычисление время перевожу в секунды
                enteredTime = (strcmp(standard, "unix") == 0) ? second : mktime(&currentDate) - timezone;

                // Difftime отнимает время друг от друга. Модуль, чтобы время не было отрицательным.
                interval = abs(difftime(enteredTime, currentTime));
                date = *gmtime(&interval);

                printf("Промежуток между датами:\nГодов: %d; суток: %d; часов: %d; минут: %d; секунд: %d\n",
                       date.tm_year - 70, date.tm_mday - 1, date.tm_hour, date.tm_min, date.tm_sec);
                break;
            }

                // Выбрать дату для работы из массива
            case (chooseDate):
```

Продолжение листинга 1

```
            {
                int dateChoice = 0;

                if (dateBase==NULL)
                {
                    printf("Массив с датами пустой!\n");
                    break;
                }

                // Вывод всех дат
                for (int i = 0; i < objectsInArray; i++)
                    printf("%d: %s\n", i + 1, asctime(&dateBase[i]));

                printf("Введите интересующий вас номер даты:\n");
                inputNat(&dateChoice);

                // Если время в диапазоне
                if (dateChoice >= 1 && dateChoice <= objectsInArray)
                {
                    currentDate = dateBase[dateChoice - 1];
                    standard = "utc";
                }
                else
                    printf("Введите значение в дипапазоне 1-%d\n",
objectsInArray);
                break;
            }

                // Удаляет дату из массива. Принцип работы аналогичен с выбором
даты
            case (dataDel):
            {
                int dateChoice = 0;

                if (dateBase==NULL)
                {
                    printf("Массив с датами пустой!\n");
                    break;
                }

                for (int i = 0; i < objectsInArray; i++)
```

Продолжение листинга 1

```
                printf("%d: %s\n", i + 1, asctime(&dateBase[i]));

            printf("Введите интересующий вас номер даты:\n");
            inputNat(&dateChoice);

            if (dateChoice >= 1 && dateChoice <= objectsInArray)
            {
                objectsInArray--;
                memcpy(dateBase,    deleteDate(dateBase,    objectsInArray,
dateChoice-1),
                        sizeof(struct tm) * objectsInArray);
                printf("Удаление произошло успешно\n");
            }
            else
                printf("Введите    значение    в    дипапазоне    1-%d\n",
objectsInArray);
            break;
        }

            // Вывод даты
        case(dataOutput):
        {
            for (int i = 0; i < objectsInArray; i++)
                printf("%s\n", asctime(&dateBase[i]));
            break;
        }

            // Отсортировывает массив по введенным пользователем ключу
        case (sort):
        {
            int order = 0;
            char* field = NULL;
            struct tm* tempBase;

            printf("По какому полю вы хотите произвести сортировку?\n"
                   "Доступные поля: seconds, minutes, hours, monthday,
weekday, month, year:\n");
            inputString(&field);
            printf("Вы хотите произвести сортировку по возрастанию или
убыванию?(1/0)\n");
            inputNat(&order);
```

Продолжение листинга 1

```
                if (order != 1 && order != 0)
                {
                    printf("Введите 1 или 0!");
                    break;
                }
                tempBase = sortArray(dateBase, objectsInArray, field, order);

                if (tempBase == NULL)
                {
                    printf("Введите правильное значение поля!\n");
                    break;
                }

                dateBase = tempBase;

                printf("Данные успешно отсортированы.\n");
                break;
            }

                // Фильтрует массив по одному или нескольким критериям
            case (filterArray):
            {
                char* keyFilterChoice = NULL;

                struct filterOutput out;

                if (dateBase==NULL)
                {
                    printf("База данных пуста!\n");
                    break;
                }

                printf("По каким полям вы хотите произвести сортировку?\n"
                       "Доступные поля: seconds, minutes, hours, monthday,
weekday, month, year:\n"
                       "Поля перечислять через запятую. Пример фильтрации:
year>2000,month=3\n");
                inputString(&keyFilterChoice);

                out = filter(dateBase, objectsInArray, keyFilterChoice);
```

Продолжение листинга 1

```
                if (out.error == 1) // Значение 100 - неправильный ключ для
фильтрации
                {
                    printf("Введите правильное условие фильтрации!\n");
                    break;
                }
                else if (out.dateArray == NULL) // NULL - условию ничего не
удовлетворяет
                {
                    printf("По данному условию ничего не найдено.\n");
                    break;
                }

                memcpy(dateBase, out.dateArray, sizeof(struct tm) * out.length);
                objectsInArray = out.length;
                for (int i = 0; i < objectsInArray; i++)
                    printf("%s\n", asctime(&dateBase[i]));
                break;
            }

           // Загрузить информацию из файла в массив
            case (loadData):
            {
                file = fopen("datebase.txt","r");

                if (file==NULL)
                {
                    printf("Файл отсутствует. Он будет создан автоматически.
Пожалуйста, повторите попытку\n");
                    file = fopen("datebase.txt","w");
                    fclose(file);
                    break;
                }

                // Первым значением в файле лежит количество переменных в массиве
                fread(&objectsInArray, sizeof(int), 1, file);
                dateBase = realloc(dateBase, sizeof(struct tm) * objectsInArray);
                fread(dateBase, sizeof(struct tm), objectsInArray, file);

                fclose(file);
```

Продолжение листинга 1

```
                printf("Данные успешно загружены.\n");
                break;
            }


                // Загрузить информацию из массива в файл
            case (dataSave):
                {
                    // Первым значением записываю длину массива
                    file = fopen("datebase.txt", "w");
                    fwrite(&objectsInArray, sizeof(int), 1, file);
                    fclose(file);

                    // Вторым значением дополняю самим массивом
                    file = fopen("datebase.txt", "a");
                    fwrite(dateBase, sizeof(struct tm), objectsInArray, file);
                    fclose(file);

                    printf("Дата была автоматически преобразована в формат UTC и
сохранена в файл.\n");
                    break;
                }

            case (exitProg):
            {
                free(dateBase);
                exit(1);
                break;
            }
        }

    }while (1);
}
```

5 Результат

Ниже представлен скриншот с консольным выводом.

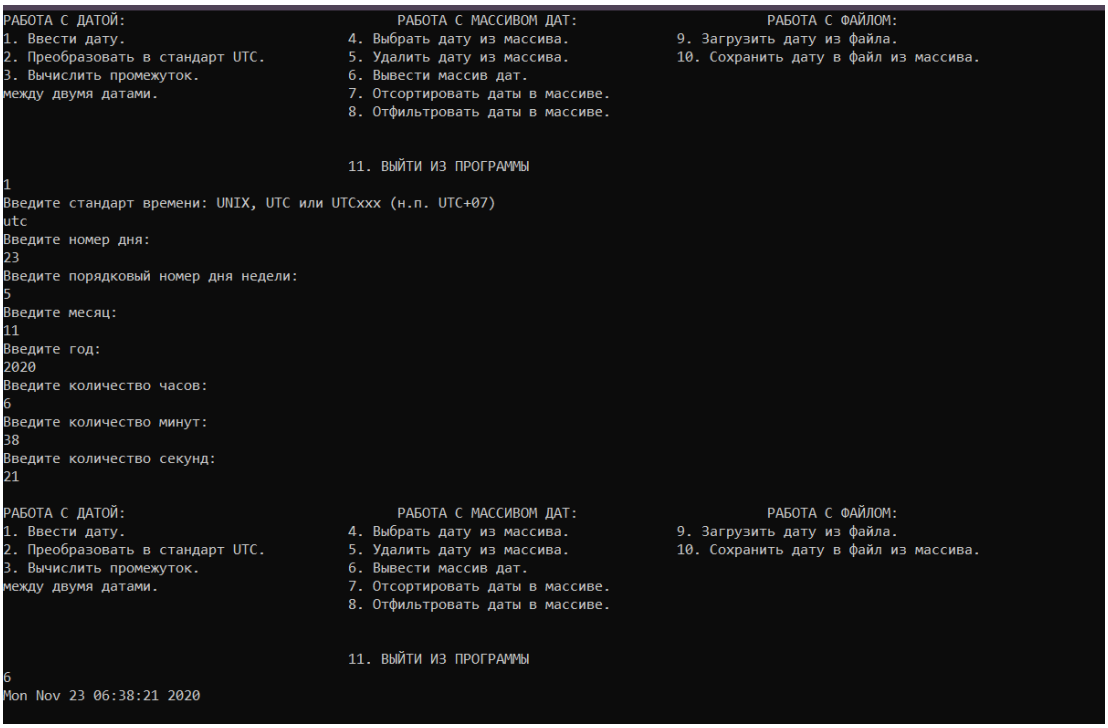

Рисунок 1 – Ввод даты с клавиатуры

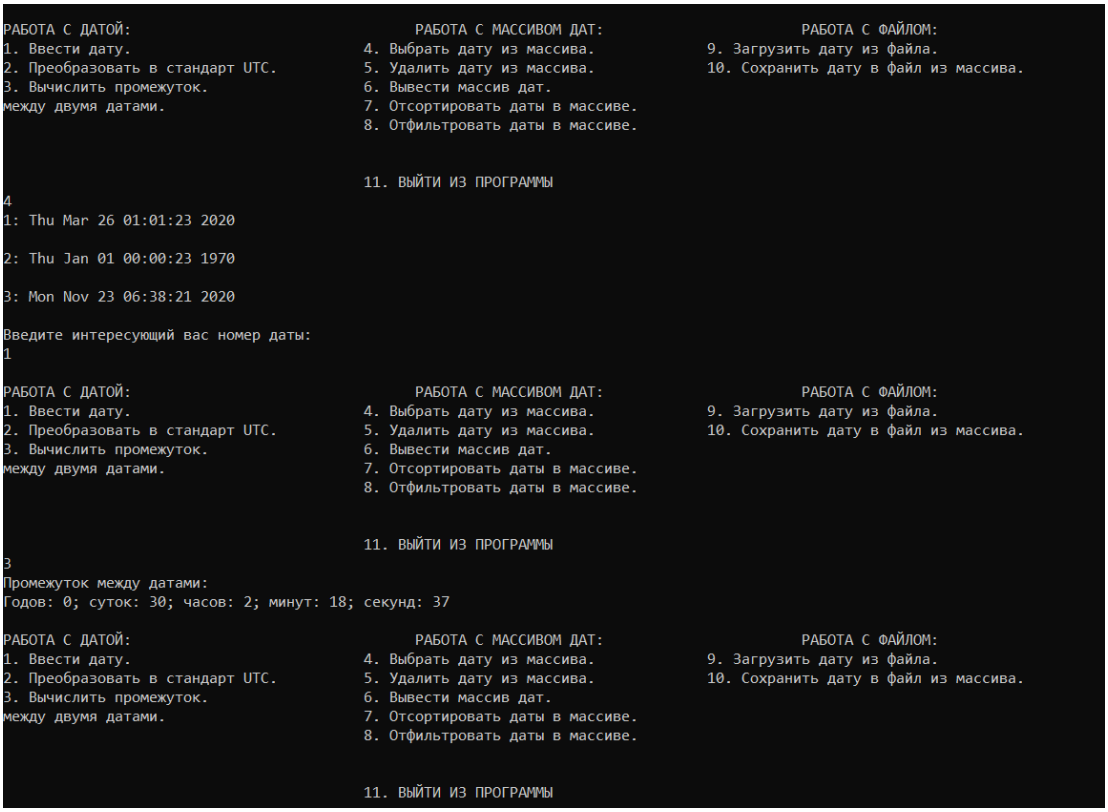

Рисунок 2 – Выбор времени и вычисление промежутка

```
7
По какому полю вы хотите произвести сортировку?
Доступные поля: seconds, minutes, hours, monthday, weekday, month, year:
year
Вы хотите произвести сортировку по возрастанию или убыванию?(1/0)
0
Данные успешно отсортированы.

РАБОТА С ДАТОЙ:                    РАБОТА С МАССИВОМ ДАТ:                    РАБОТА С ФАЙЛОМ:
1. Ввести дату.                    4. Выбрать дату из массива.               9. Загрузить дату из файла.
2. Преобразовать в стандарт UTC.   5. Удалить дату из массива.               10. Сохранить дату в файл из массива.
3. Вычислить промежуток.           6. Вывести массив дат.
между двумя датами.                7. Отсортировать даты в массиве.
                                   8. Отфильтровать даты в массиве.


                                   11. ВЫЙТИ ИЗ ПРОГРАММЫ
6
Tue Jan 01 01:01:01 2019

Fri Feb 13 23:31:30 2009

Thu Apr 19 04:25:21 2001
```

Рисунок 3 – Сортировка

```
                                   11. ВЫЙТИ ИЗ ПРОГРАММЫ
6
Fri Feb 13 23:31:30 2009

Sat Oct 09 07:16:58 2010

Tue Jan 01 01:01:01 2019


РАБОТА С ДАТОЙ:                    РАБОТА С МАССИВОМ ДАТ:                    РАБОТА С ФАЙЛОМ:
1. Ввести дату.                    4. Выбрать дату из массива.               9. Загрузить дату из файла.
2. Преобразовать в стандарт UTC.   5. Удалить дату из массива.               10. Сохранить дату в файл из массива.
3. Вычислить промежуток.           6. Вывести массив дат.
между двумя датами.                7. Отсортировать даты в массиве.
                                   8. Отфильтровать даты в массиве.


                                   11. ВЫЙТИ ИЗ ПРОГРАММЫ
8
По каким полям вы хотите произвести сортировку?
Доступные поля: seconds, minutes, hours, monthday, weekday, month, year:
Поля перечислять через запятую. Пример фильтрации: year>2000,month=3
year>2009, seconds=58
Sat Oct 09 07:16:58 2010
```

Рисунок 4 – Фильтрация

```
5
1: Fri Feb 13 23:31:30 2009

2: Sat Oct 09 07:16:58 2010

3: Tue Jan 01 01:01:01 2019

Введите интересующий вас номер даты:
2
Удаление произошло успешно

РАБОТА С ДАТОЙ:                    РАБОТА С МАССИВОМ ДАТ:                    РАБОТА С ФАЙЛОМ:
1. Ввести дату.                    4. Выбрать дату из массива.               9. Загрузить дату из файла.
2. Преобразовать в стандарт UTC.   5. Удалить дату из массива.               10. Сохранить дату в файл из массива.
3. Вычислить промежуток.           6. Вывести массив дат.
между двумя датами.                7. Отсортировать даты в массиве.
                                   8. Отфильтровать даты в массиве.


                                   11. ВЫЙТИ ИЗ ПРОГРАММЫ
6
Fri Feb 13 23:31:30 2009

Tue Jan 01 01:01:01 2019
```

Рисунок 5 – Удаление

6 Выводы

По окончании работ были выполнены следующие задачи:

1) изучены основные принципы в построении и работе со структурами;

2) проведены сложные операции над структурами, в том числе и запись их в массив;

3) реализована программа по заданию.