Бюджетное учреждение высшего образования   
Ханты-Мансийского автономного округа   
«Сургутский государственный университет»

Политехнический институт

Кафедра автоматики и компьютерных систем

**Отчет**

по лабораторной работе № 7 «Линейные алгоритмы»

по дисциплине «Программирование и основы алгоритмизации»

Выполнил: Скорба Р. В.

студент группы 609-11

Проверил: Гришмановский П. В.

доцент кафедры автоматики и компьютерных систем

Сургут

2023 г.

**Цель работы:** Закрепление теоретических знаний и приобретение практических навыков определения и использования таких типов данных, как структуры, объединения и перечисления.

**Задание:** Определить структуру для представления объекта в соответствии с вариантом индивидуального задания, выданным преподавателем и реализовать ввод, вывод и сравнение объектов структур. Полная реализация программы, соответствующей индивидуальному заданию, будет произведена в следующей работе.

Вариант 27. Задано расписание мероприятий, каждое из которых имеет текстовое описание и характеризуется датой (год, месяц, день) и временем (час, минута) начала и окончания. Обеспечить упорядочивание записей по времени начала, времени окончания, поиск перекрывающихся записей и максимального незанятого промежутка времени. Максимальное количество записей - не менее 25.

**Вариант 27.**

**Формальное описание задачи.**

Для решения поставленной задачи я реализовал структуру мероприятий (baza), содержащую следующие поля:

* int year\_start, year\_stop; - год начала, окончания мероприятия
* int mounth\_start, mounth\_stop; - месяц начала, окончания мероприятия
* int day\_start, day\_stop; день начала, окончания мероприятия
* int hours\_start, hours\_stop; час начала, окончания мероприятия
* int min\_start, min\_stop; минута начала, окончания мероприятия
* char des[255]; - описание мероприятия

Так же для реализации задачи нам я использовал пользовательские функции, такие как:

**void input(baza \*event)** – функция для ввода информации о мероприятии (заполнение структуры) ничего не возвращает из-за этого тип void;

**void output(baza \*event)** – функция для вывода информации о мероприятии (вывод структуры) ничего не возвращает из-за этого тип void;

**char s\_time\_start(baza \*event1, baza \*event2)** – функция возвращающая 0, если время начала первого мероприятия меньше чем второго, 1 в противном случае. Функция возвращает либо 0, либо 1 – взял **char,** так как это самый маленький из адресуемых типов;

**void s\_tStart(baza \*event1, baza \*event2);** - функция сортировки мероприятий по времени начала на основе функции «**int s\_time\_start()** ничего не возвращает из-за этого тип void;

**char s\_time\_stop(baza \*event1, baza \*event2) ,**

**void s\_tStop(baza \*event1, baza \*event2)** – работают аналогично как и предыдущие две функции за исключением что мероприятия проверяются и сортируются по времени окончания, а не по времени начала. Ничего не возвращает из-за этого тип void;

**char s\_overlapping(baza \*event1, baza \*event2)** – функция для проверки не пересекаются ли мероприятия. Функция возвращает либо 0, либо 1 – взял **char,** так как это самый маленький из адресуемых типов;

**char s\_Overlap(baza \*event1, baza \*event2)** - функция которая сначала сортирует мероприятия по времени начала, а затем в соответствующем порядке вызывает функцию 🡪 **int s\_overlapping().**Функция возвращает либо 0, либо 1 – взял **char,** так как это самый маленький из адресуемых типов**;**

**void s\_totalOverlap(baza \*event1, baza \*event2)** - функция которая выводит на экран информацию о том, пересекаются мероприятия или нет, вызвав предварительно функцию **int s\_Overlap()**, которая возвращает 0, если мероприятия не пересекаются и 1 если мероприятия пересекаются. Ничего не возвращает из-за этого тип void;

**void count\_interval(baza \*event1, baza \*event2)** – функция для подсчета интервала между мероприятиями в секундах. Ничего не возвращает из-за этого тип void;

**void s\_free\_interval(baza \*event1, baza \*event2**) – функция которая изначально проверяет не пересекаются ли мероприятия с помощью функции **s\_Overlap(),** в случае если мероприятия не пересекаются, то вызывается сортировка по времени начала **s\_time\_start(),** в последствии вызывается функция **count\_interval()** в порядке возрастания по времени начала мероприятий, с помощью которой производятся вычисления интервала между мероприятиями. Ничего не возвращает из-за этого тип void.

**Алгоритм программы:**

Для решения данной задачи необходимо:

1. Заполнить данные о мероприятиях
2. Выбрать в меню вид сортировки или задачи
3. Выполнить выбранное действие

**Листинг программы:**

#include <stdio.h>

#include <stdlib.h>

typedef struct baza{

int year\_start, year\_stop;

int mounth\_start, mounth\_stop;

int day\_start, day\_stop;

int hours\_start, hours\_stop;

int min\_start, min\_stop;

char des[256];

} baza;

void input(baza \*event);

void output(baza \*event);

char s\_time\_start(baza \*event1, baza \*event2);

void s\_tStart(baza \*event1, baza \*event2);

char s\_time\_stop(baza \*event1, baza \*event2);

void s\_tStop(baza \*event1, baza \*event2);

char s\_overlapping(baza \*event1, baza \*event2);

char s\_Overlap(baza \*event1, baza \*event2);

void s\_totalOverlap(baza \*event1, baza \*event2);

void s\_free\_interval(baza \*event1, baza \*event2);

void count\_interval(baza \*event1, baza \*event2);

int main(){

system("chcp 1251 > nul");

baza event1, event2;

int choose = 0;

do {

printf("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n");

printf("Ìåíþ - ðàáîòà ñ ìåðîïðèÿòèÿìè.\n1) Äîáàâèòü ìåðîïðèÿòèå.\n2) Ïðîñìîòðåòü òåêóùèå ìåðîïðèÿòèÿ.\n3) Ñîðòèðîâêà ïî âðåìåíè íà÷àëà\n4)"

" Ñîðòèðîâêà ïî âðåìåíè îêîí÷àíèÿ\n5) Ïîèñê ïåðåêðûâàþùèõñÿ çàïèñåé\n6) Ïîèñê ìàêñèìàëüíî íåçàíÿòîãî ïðîìåæóòêà âðåìåíè\n0) Âûõîä\nÂûáåðèòå æåëàåìîå äåéñòâèå: ");

while (1){

fflush(stdin);

scanf("%d", &choose);

if (choose >= 0 && choose < 7)

break;

else printf("\a\nÂûáåðèòå äåéñòâèå îò 0 äî 6!\n");

}

printf("\n~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n");

switch(choose) {

case 1:

input(&event1);

input(&event2);

break;

case 2:

output(&event1);

output(&event2);

break;

case 3:

s\_tStart(&event1, &event2);

break;

case 4:

s\_tStop(&event1, &event2);

break;

case 5:

s\_totalOverlap(&event1, &event2);

break;

case 6:

s\_free\_interval(&event1, &event2);

break;

}

} while (choose != 0);

return 0;

}

void input(baza \*event){

while (printf("Ââåäèòå äàòó íà÷àëà ìåðîïðèÿòèÿ dd mm yyyy: "), fflush(stdin),

scanf("%d%d%d", &event->day\_start, &event->mounth\_start, &event->year\_start),

event->day\_start < 1 || event->day\_start > 31 || event->mounth\_start < 1 ||

event->mounth\_start > 12 || event->year\_start < 2023 || event->year\_start > 2100)

printf("\a\nÂâåäèòå äàòó ñîãëàñíî óêàçàííîìó øàáëîíó è ëîãèêå!\n");

while (1) {

fflush(stdin);

printf("Ââåäèòå äàòó êîíöà ìåðîïðèÿòèÿ dd mm yyyy: ");

scanf("%d%d%d", &event->day\_stop, &event->mounth\_stop, &event->year\_stop);

if (event->day\_stop >= 1 || event->day\_stop <= 31 || event->mounth\_stop >= 1

|| event->mounth\_stop <= 12 || event->year\_stop >= 2023 || event->year\_stop <= 2100

|| event->year\_stop >= event->year\_start) {

if (event->year\_stop == event->year\_start) {

if (event->mounth\_stop == event->mounth\_start) {

if (event->day\_stop >= event->day\_start)

break;

else

printf("\a\nÂâåäèòå äàòó ñîãëàñíî óêàçàííîìó øàáëîíó è ëîãèêå!\n");

}

else {

if (event->mounth\_stop > event->mounth\_start)

break;

else

printf("\a\nÂâåäèòå äàòó ñîãëàñíî óêàçàííîìó øàáëîíó è ëîãèêå!\n");

}

}

else {

if (event->year\_stop < event->year\_start)

printf("\a\nÂâåäèòå äàòó ñîãëàñíî óêàçàííîìó øàáëîíó è ëîãèêå!\n");

else

break;

}

}

}

while (printf("Ââåäèòå âðåìÿ íà÷àëà ìåðîïðèÿòèÿ hh mm: "), fflush(stdin),

scanf("%d%d", &event->hours\_start, &event->min\_start),

event->hours\_start < 0 || event->hours\_start > 23 || event->min\_start < 0 ||

event->min\_start > 59)

printf("\a\nÂâåäèòå äàòó ñîãëàñíî óêàçàííîìó øàáëîíó è ëîãèêå!\n");

while (printf("Ââåäèòå âðåìÿ îêîí÷àíèÿ ìåðîïðèÿòèÿ hh mm: "), fflush(stdin),

scanf("%d%d", &event->hours\_stop, &event->min\_stop),

(event->hours\_stop < 0 || event->hours\_stop > 23 || event->min\_stop < 0 ||

event->min\_stop > 59) || (event->hours\_stop == event->hours\_start &&

event->min\_stop <= event->min\_start) && (event->year\_start == event->year\_stop &&

event->mounth\_start == event->mounth\_stop && event->day\_start == event->day\_stop)

|| (event->hours\_stop < event->hours\_start) && (event->year\_start == event->year\_stop &&

event->mounth\_start == event->mounth\_stop && event->day\_start == event->day\_stop))

printf("\a\nÂâåäèòå äàòó ñîãëàñíî óêàçàííîìó øàáëîíó è ëîãèêå!\nÌåðîïðèÿòèå íå ìîæåò çàêîí÷èòñÿ ðàíüøå ÷åì íà÷àëîñü!\n");

printf("Ââåäèòå îïèñàíèå äëÿ ìåðîïðèÿòèÿ: ");

fflush(stdin);

gets(event->des);

fflush(stdin);

printf("\n\n");

}

void output(baza \*event){

printf("Äàòà ìåðîïðèÿòèÿ - %02d.%02d.%d - %02d.%02d.%d\n", event->day\_start,

event->mounth\_start, event->year\_start, event->day\_stop, event->mounth\_stop,

event->year\_stop);

printf("Âðåìÿ ìåðîïðèÿòèÿ: %02d:%02d - %02d:%02d\n", event->hours\_start, event->min\_start,

event->hours\_stop, event->min\_stop);

printf("Îïèñàíèå: %s\n\n", event->des);

}

char s\_time\_start(baza \*event1, baza \*event2){

if (event1->year\_start > event2->year\_start) return 1;

if (event1->year\_start < event2->year\_start) return 0;

else {

if (event1->mounth\_start > event2->mounth\_start) return 1;

if (event1->mounth\_start < event2->mounth\_start) return 0;

else{

if (event1->day\_start > event2->day\_start) return 1;

if (event1->day\_start < event2->day\_start) return 0;

else{

if(event1->hours\_start > event2->hours\_start) return 1;

if(event1->hours\_start < event2->hours\_start) return 0;

else{

if (event1->min\_start > event2->min\_start) return 1;

else return 0;

}

}

}

}

}

void s\_tStart(baza \*event1, baza \*event2){

printf("Ñîðòèðîâêà ïî âðåìåíè íà÷àëà ìåðîïðèÿòèÿ:\n\n");

if (s\_time\_start(event1, event2) == 1){

output(event2);

output(event1);

}

else{

output(event1);

output(event2);

}

}

char s\_time\_stop(baza \*event1, baza \*event2){

if (event1->year\_stop > event2->year\_stop) return 1;

if (event1->year\_stop < event2->year\_stop) return 0;

else {

if (event1->mounth\_stop > event2->mounth\_stop) return 1;

if (event1->mounth\_stop < event2->mounth\_stop) return 0;

else{

if (event1->day\_stop > event2->day\_stop) return 1;

if (event1->day\_stop < event2->day\_stop) return 0;

else{

if(event1->hours\_stop > event2->hours\_stop) return 1;

if(event1->hours\_stop < event2->hours\_stop) return 0;

else{

if (event1->min\_stop > event2->min\_stop) return 1;

else return 0;

}

}

}

}

}

void s\_tStop(baza \*event1, baza \*event2){

printf("Ñîðòèðîâêà ïî âðåìåíè îêîí÷àíèÿ ìåðîïðèÿòèÿ:\n\n");

if (s\_time\_stop(event1, event2) == 1) {

output(event2);

output(event1);

}

else {

output(event1);

output(event2);

}

}

char s\_overlapping(baza \*event1, baza \*event2){

if (event1->year\_stop > event2->year\_start) return 1;

if (event1->year\_stop < event2->year\_start) return 0;

else {

if (event1->mounth\_stop > event2->mounth\_start) return 1;

if (event1->mounth\_stop < event2->mounth\_start) return 0;

else{

if (event1->day\_stop > event2->day\_start) return 1;

if (event1->day\_stop < event2->day\_start) return 0;

else{

if(event1->hours\_stop > event2->hours\_start) return 1;

if(event1->hours\_stop < event2->hours\_start) return 0;

else{

if (event1->min\_stop > event2->min\_start) return 1;

else return 0;

}

}

}

}

}

char s\_Overlap(baza \*event1, baza \*event2){

if(s\_time\_start(event1, event2) == 1) {

if(s\_overlapping(event2, event1) == 0) return 0;

else return 1;

}

else {

if (s\_overlapping(event1, event2) == 0) return 0;

else return 1;

}

}

void s\_totalOverlap(baza \*event1, baza \* event2){

if (s\_Overlap(event1, event2) == 0)

printf("Ìåðîïðèÿòèÿ íå ïåðåñåêàþòñÿ\n\n");

else

printf("Ìåðîïðèÿòèÿ ïåðåñåêàþòñÿ\n\n");

}

void count\_interval(baza \*event1, baza \*event2){

int min1 = 0, min2 = 0, h1= 0, h2 = 0, d1 = 0, d2 = 0, m1 = 0, m2 = 0, y1 = 0, y2 = 0;

int total = 0;

int mounth[13] = {0, 31, 59, 90, 120, 151, 181, 212, 243, 273, 304, 334, 365};

int i = 0, days = 0;

m1 = event1->mounth\_stop - 1;

m2 = event2->mounth\_start - 1;

y1 = event1->year\_stop - 1;

y2 = event2->year\_start - 1;

min1 = event1->min\_stop \* 60;

min2 = event2->min\_start \* 60;

h1 = event1->hours\_stop \* 60 \* 60;

h2 = event2->hours\_start \* 60 \* 60;

d1 = event1->day\_stop;

d2 = event2->day\_start;

if (event1->year\_stop % 4 == 0 || (event1->year\_stop % 100 == 0) && (event1->year\_stop % 400 == 0)){

if (m1 > 2) {

m1 = mounth[m1] + 1;

}

else m1 = mounth[m1];

}

else m1 = mounth[m1];

if (event2->year\_start % 4 == 0 || (event2->year\_start % 100 == 0) && (event2->year\_start % 400 == 0)){

if (m2 > 2) {

m2 = mounth[m2] + 1;

}

else m2 = mounth[m2];

}

else m2 = mounth[m2];

y1 = (y1 \* 365) + ((y1 / 4) - (y1 / 100) + (y1 / 400));

y2 = (y2 \* 365) + ((y2 / 4) - (y2 / 100) + (y2 / 400));

days = (y2 + m2 + d2) - (y1 + m1 + d1);

total = abs(days \* 24 \* 3600 + h2 + min2) - (h1 + min1);

printf("Ìàêñèìàëüíîå âðåìÿ ìåæäó ìåðîïðèÿòèÿìè: %d ñåêóíä\n\n", total);

}

void s\_free\_interval(baza \*event1, baza \*event2){

if (s\_Overlap(event1, event2) == 0) {

if (s\_time\_start(event1, event2) == 1) count\_interval(event2, event1);

else count\_interval(event1, event2);

}

else printf("Ó ìåðîïðèÿòèé íåò ñâîáîäíîãî èíòåðâàëà\n\n");

}

**Пояснение к программе:**

Для стабильной работы в каждом вводе учтена проверка на корректность введенных данных.

Обратимся снова к полям содержащихся в структуре Baza:

* int year\_start, year\_stop;
* int mounth\_start, mounth\_stop;
* int day\_start, day\_stop;
* int hours\_start, hours\_stop;
* int min\_start, min\_stop;
* char des[255];

Тип **int** был выбран, потому что это наиболее подходящий тип для хранения и работы с данными, все данные хранятся и записываются в целочисленном виде.

Тип **char** выбран для хранения описания мероприятия, так как это символьный тип.

Функции: *input, output, s\_tStart, s\_tStop, s\_totalOverlap, s\_free\_interval*ничего не возвращают и служат для заполнения, сортировки и вывода на экран структуры, поэтому выбран тип **void**.

Функции: *s\_time\_start, s\_time\_stop, s\_Overlap, s\_overlapping* компараторы, возвращающие либо 0, либо 1, выбран тип **char**, как самый наименьший адресный тип, занимающий всего 8 бит.

Функция *count\_free\_interval* – вспомогательная функция для работы с элементами структур, ничего не возвращающая, поэтому тип **void**.

Перед вводами и выводами выводится сообщения для удобства пользователя. Ввод и вывод организован при помощи стандартных функций scanf() и printf(), для этого подключен заголовочный файл stdio.h.

Заголовочный файл stdlib.h необходим для использования функции system(), чтобы приостановить выполнение программы перед завершением (выполняется команда «pause» операционной системы).

Для реализации меню пользователя, я использовал конструкцию switch {case: }

**Вывод:**

В ходе выполнения лабораторной работы №7 были закреплены теоретические знания и приобретены практические навыки определения и использования таких типов данных, как структуры, объединения и перечисления.