ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ТЕЛЕКОММУНИКАЦИЙ И ИНФОРМАТИКИ»

ФАКУЛЬТЕТ ИНФОРМАТИКИ И ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

Кафедра вычислительных систем

ОТЧЁТ
по лабораторной работе №6
по дисциплине «Моделирование»
на тему «Реализация монитора событий»

Выполнил:
студент группы ИВ-222
Терешков Р. В.

Проверил:
д.т.н., доцент
Родионов А. С.

Новосибирск – 2016

Задание

В рамках данной лабораторной работы необходимо реализовать монитор событий для модели, заданной следующим образом: события разделены на 2 случая. Первый случай (`A`) заключается в инкрементировании переменной `N` (функция `inc()`) с заданной интенсивностью λ, а второй (`B`) — её декрементировании (функция `dec()`).

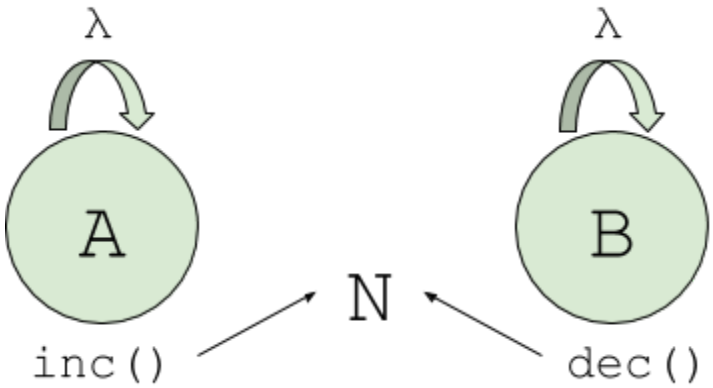

Рис. 1 — Модель для проектирования

Входные данные: λ = 1.2; `N` = 0.

Результатом работы также является подсчёт максимального и минимального значения счётчика и количество случаев, при котором счётчик был равен нулю.

Ход работы

В ходе выполнения лабораторной работы была реализована программа на языке программирования C, которая осуществляет мониторинг смоделированных событий. Каждое из событий помещается в двусвязный список — календарь событий. Структура календаря, состоящего из трёх событий, проиллюстрирована ниже:

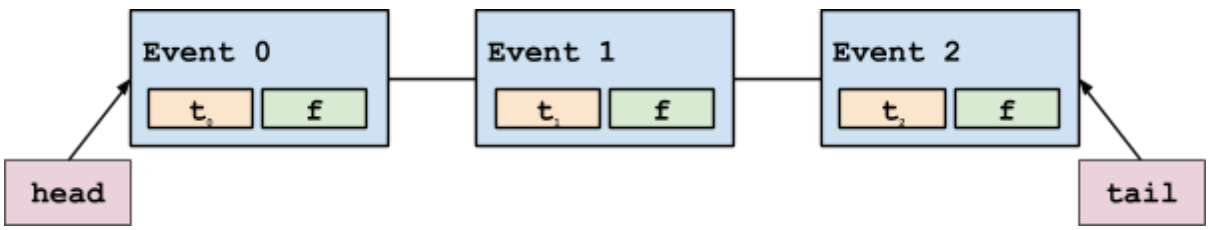

Рис. 2 — Пример календаря, состоящего из трёх событий

Причём $t_0 \leq t_1 \leq t_2$, т.е. события в календаре упорядочены в порядке возрастания по параметру `t`, обозначающего время на выполнение этого события. Поле `f` отвечает за принадлежность данного события к определённому типу: инкремент счётчика (`f` = 1) или его декремент (`f` = 0).

Генерирование величины `t` происходит по экспоненциальному закону распределения вероятности с заданным параметром λ:

$$x = -(\ln(1 - \xi) / \lambda).$$

Результаты моделирования

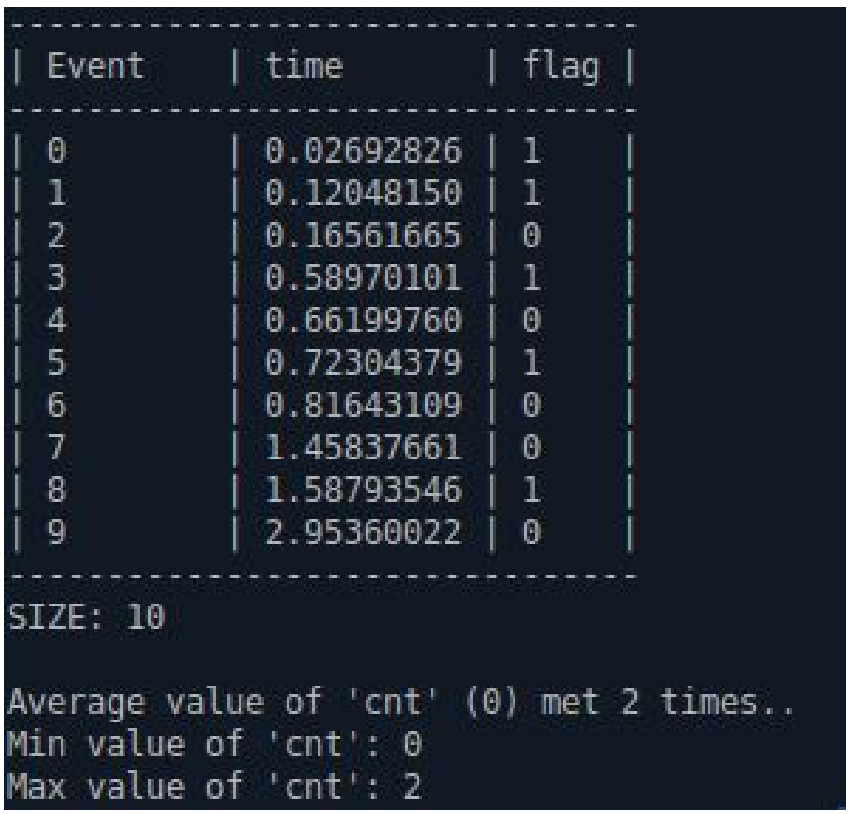

Рис. 3 — Пример вывода функции `schedule_print()`

Таблица 1 — Результаты моделирования для разного количества событий в календаре

Номер теста	Количество событий	Переменная равна нулю	Минимальное значение	Максимальное значение
1	10	2	0	2
2	100	2	-13	5
3	1000	24	-39	9
4	10000	193	-66	36
5	100000	121	-22	446

Листинг программы

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

#define N 10000

int cnt = 0;

int iimax(int a, int b) { return ((a > b) ? a : b); }
int iimin(int a, int b) { return ((a < b) ? a : b); }

void swap(double *a, double *b) { double tmp = *a; *a = *b; *b = tmp; }

struct event {
	double t;
	int f;

	struct event *next;
	struct event *prev;
};

struct schedule {
	int size;

	struct event *head;
	struct event *tail;
};

void schedule_init(struct schedule *s)
{
	s->size = 0;
	s->head = s->tail = NULL;
}

void schedule_print(struct schedule *s)
{
	struct event *tmp = s->head;

	printf("---------------------------------\n");
	printf("| Event    | time       | flag |\n");
	printf("---------------------------------\n");

	int i = 0;

	while (tmp && s->size) {
		printf("| %-9d| %-11.08lf| %-5d|\n", i, tmp->t, tmp->f);
		tmp = tmp->next;
		++i;
	}

	printf("---------------------------------\n");
}

int schedule_add(struct schedule *s, struct event *e)
{
	if (s->size == N) {
		printf("LUL cannot insert this event.\n");
```

```
        return -1;
}

struct event *tmp;
struct event *add;

tmp = s->head;

add = (struct event *) malloc(sizeof(struct event));
add->t = e->t;
add->f = e->f;

if (!s->size) {
        add->next = NULL;
        add->prev = NULL;
        s->head = s->tail = add;
} else if (s->size == 1) {
        if (add->t > tmp->t) {
                add->next = NULL;
                add->prev = tmp;

                tmp->next = add;
                tmp->prev = NULL;

                s->head = tmp;
                s->tail = add;
        } else {
                add->next = tmp;
                add->prev = NULL;

                tmp->next = NULL;
                tmp->prev = add;

                s->head = add;
                s->tail = tmp;
        }
} else {
        while (tmp) {
                if ((add->t < tmp->t) && (tmp == s->head)) {
                        add->next = tmp;
                        add->prev = NULL;

                        tmp->prev = add;

                        s->head = add;
                        break;
                } else if (add->t < tmp->t) {
                        add->next = tmp;
                        add->prev = tmp->prev;

                        tmp->prev->next = add;
                        tmp->prev = add;
                        break;
                } else if (tmp == s->tail) {
                        add->next = NULL;
                        add->prev = tmp;

                        tmp->next = add;

                        s->tail = add;
                        break;
                } else tmp = tmp->next;
```

```
                }
        }

        ++s->size;

        return 0;
}

void inc() { ++cnt; }
void dec() { --cnt; }

double drand() { return rand() / (double) RAND_MAX; }

double distr(double ksi, double lambda) { return -1.0 * (log(1.0 - ksi) / lambda); }

int main()
{
        srand((unsigned)time(NULL));

        int zero = 0, max = 0, min = 0;

        struct event *list1 = (struct event *) malloc(sizeof(struct event) * (N / 2));
        struct event *list2 = (struct event *) malloc(sizeof(struct event) * (N / 2));

        struct schedule *sched = (struct schedule *) malloc(sizeof(struct schedule) * N);

        schedule_init(sched);

        for (int i = 0; i < N / 2; ++i) {
                double ksi = drand();
                list1[i].t = distr(ksi, 1.2);
                list1[i].f = 1;
        }

        for (int i = 0; i < N / 2; ++i) {
                double ksi = drand();
                list2[i].t = distr(ksi, 1.2);
                list2[i].f = 0;
        }

        for (int i = 0; i < N / 2; ++i)
                schedule_add(sched, &list1[i]);

        for (int i = 0; i < N / 2; ++i)
                schedule_add(sched, &list2[i]);

        schedule_print(sched);

        struct event *tmp = sched->head;

        while (tmp && sched->size) {
                if (tmp->f) inc();
                else dec();

                if (!cnt) ++zero;

                min = iimin(min, cnt);
                max = iimax(max, cnt);

                tmp = tmp->next;
        }
```

```
        printf("SIZE: %d\n\n", sched->size);

        printf("Average value of 'cnt' (0) met %d times..\n", zero);
        printf("Min value of 'cnt': %d\n", min);
        printf("Max value of 'cnt': %d\n", max);

        free(list1);
        free(list2);
        free(sched);

        return 0;
}
```