Отчет
Дз10
Пирогов Даниил
БПИ-245

Использованы POSIX потоки, мьютексы и условные переменные
Общий буфер реализован как кольцевая очередь, при этом источники и сумматоры ждут, если буфер заполнен, а монитор ждет, если в буфере меньше двух чисел
Поток-монитор извлекает пары чисел и создаёт для них потоки-сумматоры
Количество активных сумматоров отслеживается счётчиком, чтобы корректно определить момент завершения вычислений
Программа завершается, когда все источники отработали, все сумматоры завершены и в буфере осталось ровно одно число

Исходный код:

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <unistd.h>
#include <time.h>
#include <stdbool.h>
#include <stdarg.h>

#define SRC_N 100
#define BUF_SZ 200

static pthread_mutex_t m = PTHREAD_MUTEX_INITIALIZER;
static pthread_cond_t ne = PTHREAD_COND_INITIALIZER;
static pthread_cond_t nf = PTHREAD_COND_INITIALIZER;
static pthread_mutex_t pm = PTHREAD_MUTEX_INITIALIZER;
static int q[BUF_SZ];
static int hd = 0, tl = 0, cnt = 0;
static int gen = 0;
static int sum = 0;
static int act = 0;
static bool fin = false;

typedef struct { int id; unsigned int s; } src_arg;
typedef struct { int id, a, b; unsigned int s; } sum_arg;

static void plog(const char* fmt, ...) {
    va_list ap;
    pthread_mutex_lock(&pm);
    va_start(ap, fmt);
    vprintf(fmt, ap);
    va_end(ap);
    fflush(stdout);
    pthread_mutex_unlock(&pm);
}

static void q_put(int v) {
    while (cnt == BUF_SZ) pthread_cond_wait(&nf, &m);
    q[tl] = v;
    tl = (tl + 1) % BUF_SZ;
    cnt++;
```

```
    pthread_cond_broadcast(&ne);
}

static void q_get2(int* a, int* b) {
    while (cnt < 2 && !fin) pthread_cond_wait(&ne, &m);
    if (fin && cnt < 2) { *a = *b = 0; return; }
    *a = q[hd]; hd = (hd + 1) % BUF_SZ; cnt--;
    *b = q[hd]; hd = (hd + 1) % BUF_SZ; cnt--;
    pthread_cond_broadcast(&nf);
}

static void* src_th(void* p) {
    src_arg* x = (src_arg*)p;
    int id = x->id;
    unsigned int s = x->s;
    int d = 1 + (int)(rand_r(&s) % 7u);
    sleep((unsigned)d);
    int v = id + 1;
    pthread_mutex_lock(&m);
    q_put(v);
    gen++;
    plog("Источник %03d Сгенерировал %d (%d сек)\n", id, v, d);
    plog("Буфер %d : элементов в буфере: %d\n", v, cnt);
    pthread_mutex_unlock(&m);
    free(x);
    return NULL;
}

static void* sum_th(void* p) {
    sum_arg* x = (sum_arg*)p;
    int id = x->id, a = x->a, b = x->b;
    unsigned int s = x->s;
    int d = 3 + (int)(rand_r(&s) % 4u);
    plog("Сумматор %03d Старт: %d + %d (%d сек)\n", id, a, b, d);
    sleep((unsigned)d);
    int r = a + b;
    pthread_mutex_lock(&m);
    q_put(r);
    sum++;
    plog("Сумматор %03d Готово: %d + %d = %d\n", id, a, b, r);
    plog("Буфер %d : элементов в буфере: %d\n", r, cnt);
    act--;
    pthread_cond_broadcast(&ne);
    pthread_mutex_unlock(&m);
    free(x);
    return NULL;
}

static void* mon_th(void* p) {
    (void)p;
    int sid = 0;
    while (1) {
        pthread_mutex_lock(&m);
        int a, b;
        q_get2(&a, &b);
        if (fin && cnt < 2) { pthread_mutex_unlock(&m); break; }
        plog("Взяли пару: %d и %d : осталось в буфере: %d\n", a, b, cnt);
        plog("Сгенерировано: %d/%d + суммирований: %d + активных сумматоров: %d\n",
            gen, SRC_N, sum, act);
        sum_arg* x = (sum_arg*)malloc(sizeof(sum_arg));
```

```
        x->id = sid++;
        x->a = a;
        x->b = b;
        x->s = (unsigned)time(NULL) ^ (unsigned)(x->id * 2654435761u);
        act++;
        pthread_t t;
        int rc = pthread_create(&t, NULL, sum_th, x);
        if (rc == 0) {
            pthread_detach(t);
        } else {
            act--;
            q_put(a);
            q_put(b);
            free(x);
            plog("ОШИБКА: не удалось создать сумматор (rc=%d)\n", rc);
        }
        pthread_mutex_unlock(&m);
    }
    return NULL;
}

int main() {
    pthread_t s[SRC_N], mon;
    plog("Да начнутся гонки!\n");
    pthread_create(&mon, NULL, mon_th, NULL);

    for (int i = 0; i < SRC_N; i++) {
        src_arg* a = (src_arg*)malloc(sizeof(src_arg));
        a->id = i;
        a->s = (unsigned)time(NULL) ^ (unsigned)(i * 1103515245u + 12345u);
        pthread_create(&s[i], NULL, src_th, a);
    }

    for (int i = 0; i < SRC_N; i++) pthread_join(s[i], NULL);
    pthread_mutex_lock(&m);
    while (!(gen == SRC_N && act == 0 && cnt == 1)) {
        pthread_cond_wait(&ne, &m);
    }
    int res = q[hd];
    fin = true;
    pthread_cond_broadcast(&ne);
    pthread_mutex_unlock(&m);
    pthread_join(mon, NULL);
    plog("\nHappy end\nФинальный результат: %d\nСгенерировано чисел: %d\nВыполнено
суммирований: %d\n",
         res, gen, sum);
    pthread_mutex_destroy(&m);
    pthread_cond_destroy(&ne);
    pthread_cond_destroy(&nf);
    pthread_mutex_destroy(&pm);
    return 0;
}
```