ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук
Образовательная программа бакалавриата «Программная инженерия»

Домашняя работа №4 по учебному курсу "Архитектура вычислительных систем"

Титульный лист

Вариант 19

Исполнитель:
студент группы БПИ191 А.А.Новоселов

Москва 2020

1 Текст задания вариант 19

У одной очень привлекательной студентки есть N поклонников.Традиционно в день св. Валентина очень привлекательная студенткапроводит романтический вечер с одним из поклонников. Счастливыйизбранник заранее не известен. С утра очень привлекательная студенткаполучает N «валентинок» с различными вариантами романтического вечера.Выбрав наиболее заманчивое предложение, студентка извещает счастливчикао своем согласии, а остальных – об отказе. Требуется создать многопоточноеприложение, моделирующее поведение студентки. При решенииииспользовать парадигму «клиент-сервер» с активным ожиданием.

2 Решение задачи

Я додумал некоторые моменты условия задачи, т.к. они не описаны явно. Например, студентка выбирает поклонника по кол-ву латинских букв(не важно заглавных или строчных) в его записке, а сама записка является строкой. Так же я не понял как извещать потоки(т.е. передавать какие то данные не через буффер) о чем-то, например об отказе студентки, поэтому я запоминал в глобальной переменной уникальный номер студент с лучшей запиской. После завершения потока сервера(т.е. когда он обработает все валентинки) каждый клиент(поклонник) сверяет свой номер с номером лучшей валентинки и выводит сообщение в консоль, если его номер совпал.

Программа построена на основе клиент-серверной модели поведения, где несколько потоков выдают запросы серверу, а сервер ожидает их и отвечает. Важным при решении было реализация активного ожидания для сервера. Большая часть программы разделена на 2 функции(которые выполняют создаваемые потоки): клиент и сервер. Так же есть 2 вспомогательные функции. В основной программе(функции main) считываются входные данные, создаются клинетские и серверный поток.

Для создания активного ожидания у сервера использовались условные переменные и 2 мютекса, для чтения и записи данных в буфер. При решении использовалась стандартная библиотека pthread.

Для обмена данными между потоками использовались 2 переменные(буфферы), в которые клиент вписывал свое послание и свой уникальный номер, а сервер их читал.

Так же для удобства тестирования в программе есть функция генерации случайной строки(из символов с ascii кодом от 32 до 126) случайной длины из диапазона [5;15].

3 Текст программы

```cpp
#include <iostream>
#include <cstdlib>
#include <thread>
#include <mutex>
#include <condition_variable>
#include <vector>
#include <ctime>

#define _rand(min, max) ( rand() % ((max) - (min) + 1) + (min) ) //возвращает случайное число из диапазона

std::mutex read_mutex, write_mutex, print_mutex; // мютексы для лока блоков кода
std::condition_variable condc, conds, condp; // используется для сигнализации
std::string buffer = ""; //текущая валентинка
int n = 0; //текущий номер студента
int best_valentives_note = 0; // номер лучшей валентинки
int best_valentines_note_length = 0; //лучшая длина валентинки
bool flag = false; //флаг что сервер обработал все запросы

std::string generateString()
{
    //генерирует рандомную строку рандомной длины. Начальное значение задается перед вызовом(в клиенте)
    int length = _rand(5, 15);
    std::string message = "";
    for (int i = 0; i < length; ++i)
    {
        message += (char)_rand(32, 126);
    }
    return message;
}
```

```cpp
bool check_valentines_note()
{
    int count = 0;
    for (int i = 0; i < buffer.length(); ++i)
    {

        //выбираем только символы
        if ((buffer[i] > 64 && buffer[i] < 91) || (buffer[i] > 96 && buffer[i] < 123)) //определим входит ли буква в латинский алфавит
        {
            ++count;
        }
    }

    //узнаем подходит ли больше нам эта строка
    if (count > best_valentines_note_length)
    {
        //чтобы не происходило одноверменной записи(хотя поток сервера 1, но на всякий случай проверим)
        std::unique_lock<std::mutex> write_lock([&]write_mutex);
        best_valentines_note_length = count;
        return true;
    }
    return false;
}
```

```cpp
void client(int num) {
    {
        srand(static_cast<unsigned int>(int(time(NULL)) ^ num));
        //сигналом что студентка прочитала валентинку, является освобождение буффера
        {
            //проверим мютекс
            std::unique_lock<std::mutex> locker([&]write_mutex);
            condc.wait([&]locker, _Pred:[&]()->bool {return buffer == "";});

            buffer = generateString(); //сгенерируем строку
            std::cout << "I m student number " << num + 1 << " and my message is " << buffer << std::endl; //выведем сгенерированное сообщение
            n = num; //запишем свой номер в буфер обмена
        }

        conds.notify_one();
    }

    //Ждем пока студентка прочитает все валентинки
    std::this_thread::yield();
    {
        //Ждем своей очереди чтобы узнать выбрала ли она нас
        std::unique_lock<std::mutex> locker([&]print_mutex);
        condp.wait([&]locker, _Pred:[&]()->bool {return flag;});

        if (best_valentives_note == num)
        {
            std::cout << "I am student " << num + 1 << " and i won competition! " << best_valentives_note + 1 << std::endl;
        }
        //Если нет, то грустно плачем в одиночестве(
    }
    condp.notify_all();
}
```

```cpp
void server(int param)
{
    for (int i = 0; i < param; ++i)
    {
        {
            std::unique_lock<std::mutex> locker([&]read_mutex);

            //сигналом что кто-то отправил валентинку является непустой буффер
            conds.wait([&]locker, _Pred:[&]()->bool {return buffer != "";}); // пока кто-нибудь не запишет данные, мы ждем

            //обработаем полученную валентинку
            if (check_valentines_note())
            {
                std::unique_lock<std::mutex> locker1([&]write_mutex);
                best_valentives_note = n;
            }
        }
        {
            //очистим буффер
            std::unique_lock<std::mutex> locker([&]write_mutex);
            buffer = "";
        }

        //дождемся записи клиента
        condc.notify_one();
        std::this_thread::yield();
    }
    flag = true;
    condp.notify_one();
}
```

```cpp
int main(int argc, char* argv[])
{
    srand(static_cast<unsigned int>(time(NULL))); //зададим случайное начальное значение
    const int number_of_students = _rand(5, 15); // сгенерируем рандомное кол-во фанатов

    std::cout << "A student has " << number_of_students << " admires" << std::endl;

    std::vector<std::thread> students_treads; //храним ссылки на потоки, чтобы они не вышли из области видимости и не уничтожились раньше времени

    std::thread server_thread = std::thread([&]server, number_of_students); //создадим поток сервера

    int i = 0;
    for (; i < number_of_students; ++i)
    {
        students_treads.push_back(_Val:std::thread([&]client, i)); //заполним массив потоков
    }

    for (int i = 0; i < number_of_students; ++i)
    {
        students_treads[i].join();
    }

    server_thread.join(); //чтобы поток был безопасен для удаления при выходе из области видимости
}
```