МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ

ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ

УНИВЕРСИТЕТ им. В. Г. ШУХОВА» (БГТУ им. В.Г. Шухова)

Кафедра программного обеспечения

вычислительной техники и автоматизированных

систем

**Лабораторная работа №6**

по дисциплине: ООП

тема: «Потоки в C++»

Выполнил: студент группы ВТ-231

Масленников Д. А.

Проверили:  
Буханов Д. Г.

Белгород 2025

**Цель работы:** изучение основных возможностей потоков управления и потоков ввода-вывода. Получение навыков работы со стандартными средствами управления потоками в С++11. Знакомство с классом Thread и стандартными средствами синхронизации потоков.

**Задание:**

1. Изучить основные классы и их возможности работы с потоками в С++11.
2. Разработать программу в соответствии с вариантом задания. Программа должна содержать 2 потока Thread для реализации основного задания лабораторной работы. Вывод организовать в отдельном потоке.
3. Реализовать классы и выполнить перегрузку оператора функтора для реализации поставленной основной задачи.
4. Разработать программу в соответствии с вариантом задания (номер варианта + 3), используя API CreateThread.

Сделать выводы о проделанной работе.

**Вариант 7**

Один поток создает бегущую строку (из случайных символов), а другой вставляет или удаляет между ними случайные знаки препинания. Произвести синхронный вывод при каждой итерации. Показать выполнение работы программы в синхронном и асинхронном режимах.

Асинхронный режим  
  
#include <iostream>

#include <thread>

#include <mutex>

#include <cstdlib>

#include <ctime>

#include <unistd.h>

**class** RunningText {

private:

std::string text;

std::mutex mtx;

**bool** running;

public:

RunningText() : running(true) {

srand(time(nullptr));

}

**void** stop() {

running = false;

}

**void** generateRunningString() {

**while** (running) {

mtx.lock();

**char** c = 'a' + rand() % 26;

text += c;

**if** (text.length() > 20) {

text = text.substr(1);

}

mtx.unlock();

usleep(300000);

}

}

**void** modifyPunctuation() {

**const** std::string punctuation = ".,!?;:";

**while** (running) {

mtx.lock();

**if** (!text.empty()) {

**if** (rand() % 2 == 0) {

**size\_t** pos = rand() % (text.length() + 1);

**char** p = punctuation[rand() % punctuation.length()];

text.insert(pos, 1, p);

} **else** {

**size\_t** found = text.find\_first\_of(punctuation);

**if** (found != std::string::npos) {

text.erase(found, 1);

}

}

}

mtx.unlock();

usleep(400000);

}

}

**void** display() {

**while** (running) {

mtx.lock();

std::cout << "\rТекущая строка: " << text << "\tДлина: " << text.length();

std::cout.flush();

mtx.unlock();

usleep(200000);

}

std::cout << std::endl;

}

std::string getText() {

**return** text;

}

};

**int** main() {

RunningText rt;

std::**thread** generator(&RunningText::generateRunningString, &rt);

std::**thread** modifier(&RunningText::modifyPunctuation, &rt);

std::**thread** display(&RunningText::display, &rt);

sleep(10);

rt.stop();

generator.join();

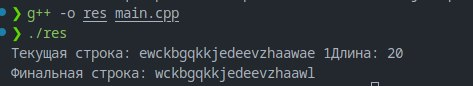
modifier.join();

display.join();

std::cout << "Финальная строка: " << rt.getText() << std::endl;

**return** 0;

}

  
  
**Синхронный режим:**  
  
  
#include <iostream>

#include <cstdlib>

#include <ctime>

#include <unistd.h>

**class** RunningText {

private:

std::string text;

**bool** running;

public:

RunningText() : running(true) {

srand(time(nullptr));

}

**void** stop() {

running = false;

}

**void** generateRunningString() {

**char** c = 'a' + rand() % 26;

text += c;

**if** (text.length() > 20) {

text = text.substr(0, 20);

stop();

}

}

**void** modifyPunctuation() {

**const** std::string punctuation = ".,!?;:";

**if** (!text.empty() && text.length() <= 20) {

**if** (rand() % 2 == 0) {

**size\_t** pos = rand() % (text.length() + 1);

**char** p = punctuation[rand() % punctuation.length()];

text.insert(pos, 1, p);

**if** (text.length() > 20) {

text = text.substr(0, 20);

stop();

}

} **else** {

**size\_t** found = text.find\_first\_of(punctuation);

**if** (found != std::string::npos) {

text.erase(found, 1);

}

}

}

}

**void** display() {

std::cout << "\rТекущая строка: " << text << " \tДлина: " << text.length();

std::cout.flush();

}

**void** runSync() {

**while** (running) {

generateRunningString();

modifyPunctuation();

display();

usleep(200000);

**if** (text.length() >= 20) {

stop();

}

}

std::cout << std::endl;

}

std::string getText() {

**return** text;

}

};

**int** main() {

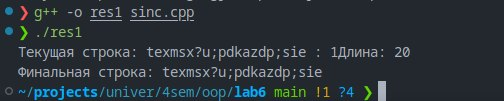
RunningText rt;

rt.runSync();

std::cout << "Финальная строка: " << rt.getText() << std::endl;

**return** 0;

}

  
  
  
  
  
  
  
  
  
**Вывод**: **Асинхронный режим повышает скорость выполнения программы за счёт параллельной обработки задач, но усложняет код из-за необходимости синхронизации потоков и управления состоянием.**