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 №9

по дисциплине: Объектно-ориентированное программирование

тема: «**Использование стандартной библиотеки шаблонов STL»**

Выполнил: ст. группы ПВ-233

Ситников Алексей Павлович

Проверил:

Белгород 2025 г.

Вариант 3 (13)

Цель работы: знакомство со стандартной библиотекой шаблонов в C++; получение навыков использования классов контейнеров, итераторов, алгоритмов.

Разработать программное обеспечение для решения следующей задачи: построение очереди обработки задач. Задачи следующего вида, создание файла, удаление файла, переименование файла, вывод файла на экран, добавление записи в файл, удаление записи из файла. Один поток берет задачу из очереди, и производит ее выполнение, другие потоки, число которых задается, динамически выполняют добавление задач в очередь. Организовать слияние очередей задач на основе времени добавление задачи.

```
#include <iostream>
#include <fstream>
#include <queue>
#include <windows.h>
#include <functional>
#include <thread>
#include <mutex>
#include <filesystem>
#include <chrono>


std::mutex m1;
std::mutex m3;
std::string inputData(int i){
    std::string temp;
    if(i == 1) {
        std::cin >> temp;
        m1.unlock();
    }
    else if(i == 2){
        std::cin.ignore();
        std::getline(std::cin, temp);
    }
    return temp;
}

int createFile(){
    m3.lock();
    std::string name;
    m1.lock();
    std::this_thread::sleep_for(std::chrono::milliseconds (500));
    std::cout << "Введите имя файла для создания\n";
    name = inputData(1);
    std::ifstream file(name);
    if (file.good()) {
        std::cerr << "Ошибка: Файл '" << name << "' уже существует!" <<
std::endl<< std::flush;
        m3.unlock();
        return 1;
    }
    std::ofstream F;
    F.open(name);
    if (!F.is_open()) {
```

```
        std::cerr << "Не удалось создать файл '" << name << "'!" <<
std::endl<< std::flush;
        m3.unlock();
        return 1;
    }
    F.close();
    std::cout << "файл "<< name << " создан"<< std::endl<< std::flush;
    m3.unlock();
    return 0;
}

int deleteFile(){
    m3.lock();
    std::string name;
    m1.lock();
    std::this_thread::sleep_for(std::chrono::milliseconds (500));
    std::cout << "Введите имя файла для удаления\n";
    name = inputData(1);
    std::ifstream file(name);
    if (!file.good()) {
        std::cerr << "Ошибка: Файл '" << name << "' не существует!" <<
std::endl << std::flush;
        m3.unlock();
        return 1;
    }
    file.close();
    if(std::remove(name.c_str())){
        std::cerr << "Ошибка: Файл '" << name << "' не был удалён!" <<
std::endl<< std::flush;
        m3.unlock();
        return 1;
    }
    std::cout << "файл "<< name << " удалён"<< std::endl<< std::flush;
    m3.unlock();
    return 0;
}

int renameFile(){
    m3.lock();
    std::string name;
    m1.lock();
    std::this_thread::sleep_for(std::chrono::milliseconds (500));
    std::cout << "Введите имя файла для переименования\n";
    name = inputData(1);
    std::ifstream file(name);
    if (!file.good()) {
        std::cerr << "Ошибка: Файл '" << name << "' не существует!" <<
std::endl<< std::flush;
        m3.unlock();
        return 1;
    }
    std::string newName;
    std::cout << "Введите новое имя файла\n";
    newName = inputData(1);

    std::ifstream f(newName);
    if (f.good()) {
        std::cerr << "Ошибка: Файл '" << newName << "' уже существует!" <<
std::endl<< std::flush;
        m3.unlock();
        return 1;
    }
    file.close();
    f.close();
```

```
    try {
        std::filesystem::rename(name, newName);
    }
    catch (std::exception &e){
        std::cerr << "Ошибка: Файл '" << name << "' не был переименован!\n"
<< std::flush;
        m3.unlock();
        return 1;
    }
    std::cout << "файл "<< name << " переименован на " << newName<<
std::endl<< std::flush;
    m3.unlock();
    return 0;
}

int printFile(){
    m3.lock();
    std::string name;
    m1.lock();
    std::this_thread::sleep_for(std::chrono::milliseconds (500));
    std::cout << "Введите имя файла для вывода\n";
    name = inputData(1);
    std::ifstream F;
    F.open(name);
    if(!F.is_open()){
        std::cerr << "Ошибка: Файл '" << name << "' не был открыт" <<
std::endl<< std::flush;
        m3.unlock();
        return 1;
    }
    std::string line;
    while (std::getline(F, line))
    {
        std::cout << line << std::endl;
    }
    F.close();
    m3.unlock();
    return 0;
}

int addDataFile(){
    m3.lock();
    std::string name;
    m1.lock();
    std::this_thread::sleep_for(std::chrono::milliseconds (500));
    std::cout << "Введите имя файла для добавления данных\n";
    name = inputData(1);
    std::ifstream F;
    F.open(name);
    if(!F.is_open()){
        std::cerr << "Ошибка: Файл '" << name << "' не существует!" <<
std::endl<< std::flush;
        m3.unlock();
        return 1;
    }
    std::ofstream file;
    file.open(name, std::ios::app);
    if(!file.is_open()){
        std::cerr << "Ошибка: Файл '" << name << "' не был открыт!" <<
std::endl<< std::flush;
        m3.unlock();
        return 1;
    }
    std::string data;
```

```
    std::cout << "Введите данные для добавления\n";
    data = inputData(2);
    file << data << std::endl<< std::flush;
    file.close();
    m1.unlock();
    F.close();
    std::cout << "данные добавлены в файл "<< name<< std::endl<< std::flush;
    m3.unlock();
    return 0;
}

int clearFile() {
    m3.lock();
    std::string name;
    m1.lock();
    std::this_thread::sleep_for(std::chrono::milliseconds (500));
    std::cout << "Введите имя файла для его отчистки\n";
    name = inputData(1);
    std::ofstream file(name, std::ios::trunc);

    if (!file.is_open()) {
        std::cerr << "Ошибка: Не удалось открыть файл '" << name << "'!" <<
std::endl;
        m3.unlock();
        return 1;
    }
    file.close();
    std::cout << "файл "<< name << " отчищен" << std::endl;
    m3.unlock();
    return 0;
}

int flag = 0;

int end(){
    flag = 1;
    std::this_thread::sleep_for(std::chrono::milliseconds (1000));
    std::cout << "end"<< std::endl;
    return 0;
}

std::mutex m2;
std::function<int()> queue_(int i, std::queue<std::function<int()>> &t,
std::function<int()> a){
    m2.lock();
    if (i == 1){
        std::function<int()> ans = t.front();
        t.pop();
        m2.unlock();
        return ans;
    }
    else if(i == 2){
        t.emplace(a);
    }
    m2.unlock();
    return nullptr;
}

int flagF = 0;

void solve(std::queue<std::function<int()>> &t){
    while (true){
        if(!t.empty() && flagF){
```

```
            std::function<int()> func = queue_(1, t, nullptr);

            func();
        }
        else if(flagF){
            if(flag == 1){
                break;
            }
        }
    }
}



std::vector<std::queue<std::function<int()>>> queues;
std::vector<std::vector<std::time_t>> times;

//в первую очередь слияние производится
void combineQueues(){
    m2.lock();
    std::queue<std::function<int()>> newQ;
    std::vector<int> index(queues.size());
    long long int min;
    int x_indx;
    while (true){
        min = LONG_LONG_MAX;
        for(int i = 0; i < queues.size(); i++){
            if(index[i] == times[i].size()){
                continue;
            }
            if(min > times[i][index[i]]){
                x_indx = i;
                min = times[i][index[i]];
            }
        }
        if(min!=LONG_LONG_MAX){
            std::function<int()> func = queues[x_indx].front();
            queues[x_indx].pop();
            newQ.push(func);
            index[x_indx]++;
        }
        else{
            break;
        }

    }
    queues[0] = std::move(newQ);
    m2.unlock();
}



std::mutex m4;
void addFunc(std::queue<std::function<int()>> &t, int i){

    while (true) {
        std::string data;
        std::this_thread::sleep_for(std::chrono::seconds(2));
        m3.lock();
        m1.lock();
        m4.lock();
        if(flagF == 1){
            m3.unlock();
            m1.unlock();
```

```
            m4.unlock();
            std::cout.flush();
            break;
        }

        std::this_thread::sleep_for(std::chrono::seconds(2));
        std::cout<<"id потока = "<< i << "\nВведите название функции\n";
        data = inputData(1);
        m3.unlock();
        if (data == "createFile") {

times[i].push_back(std::chrono::system_clock::to_time_t(std::chrono::system_c
lock::now()));
            queue_(2, t, createFile);
        } else if (data == "deleteFile") {

times[i].push_back(std::chrono::system_clock::to_time_t(std::chrono::system_c
lock::now()));
            queue_(2, t, deleteFile);
        } else if (data == "renameFile") {

times[i].push_back(std::chrono::system_clock::to_time_t(std::chrono::system_c
lock::now()));
            queue_(2, t, renameFile);
        } else if (data == "printFile") {

times[i].push_back(std::chrono::system_clock::to_time_t(std::chrono::system_c
lock::now()));
            queue_(2, t, printFile);
        } else if (data == "addDataFile") {

times[i].push_back(std::chrono::system_clock::to_time_t(std::chrono::system_c
lock::now()));
            queue_(2, t, addDataFile);
        } else if (data == "clearFile") {

times[i].push_back(std::chrono::system_clock::to_time_t(std::chrono::system_c
lock::now()));
            queue_(2, t, clearFile);
        }
        else if(data == "end"){

times[i].push_back(std::chrono::system_clock::to_time_t(std::chrono::system_c
lock::now()));
            queue_(2, t, end);
            combineQueues();
            flagF = 1;
            flag = 1;
            m4.unlock();
            break;
        }
        else{
            std::cout<<"Неизвестная функция" << std::endl;
        }
        m4.unlock();
    }
}


void setCount(int count) {
    queues.resize(count);
    times.resize(count);
}
```

```cpp
int main() {
    SetConsoleOutputCP(CP_UTF8);

    int count;
    std::cout << "Введите количество потоков\n";
    std::cin >> count;
    setCount(count);
    std::thread tr(solve, std::ref(queues[0]));
    std::vector<std::thread> treads;
    for(int i = 0; i < count; i++){
        treads.emplace_back(addFunc, std::ref(queues[i]), i);
    }



    for(std::thread &i : treads){
        i.join();
    }
    tr.join();
    return 0;
}
```

Вывод программы:

```
C:\Users\admin\CLionProjects\OOP\cmake-build-debug\OOP.exe
Введите количество потоков
2
id потока = 1
Введите название функции
createFile
id потока = 0
Введите название функции
deleteFile
id потока = 1
Введите название функции
end
Введите имя файла для создания
fff.txt
файл fff.txt создан
Введите имя файла для удаления
fff.txt
файл fff.txt удалён
end

Process finished with exit code 0
```

```
C:\Users\admin\CLionProjects\OOP\cmake-build-debug\OOP.exe
Введите количество потоков
4
id потока = 1
Введите название функции
createFile
id потока = 2
Введите название функции
createFile
id потока = 3
Введите название функции
renameFile
id потока = 0
Введите название функции
renameFile
id потока = 1
Введите название функции
addDataFile
id потока = 2
Введите название функции
clearFile
id потока = 3
Введите название функции
deleteFile
id потока = 0
Введите название функции
end
Введите имя файла для создания
file.txt
файл file.txt создан
Введите имя файла для создания
file.txt
Ошибка: Файл 'file.txt' уже существует!
Введите имя файла для переименования
file.txt
Введите новое имя файла
```

```
main.py
файл file.txt переименован на main.py
Введите имя файла для переименования
main.py
Введите новое имя файла
main.cpp
файл main.py переименован на main.cpp
Введите имя файла для добавления данных
main.cpp
Введите данные для добавления
#include <iostream>
данные добавлены в файл main.cpp
Введите имя файла для его отчистки
f.txt
файл f.txt отчищен
Введите имя файла для удаления
f.txt
файл f.txt удалён
end

Process finished with exit code 0
```

Вывод: я ознакомился со стандартной библиотекой шаблонов в C++; получил навыки использования классов контейнеров, итераторов, алгоритмов.