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-223

Игнатьев Артур Олегович

Проверил:

асс. Черников Сергей Викторович

Белгород 2024г.

**Лабораторная работа №9**

**«Использование стандартной библиотеки шаблонов STL»**

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

**Вариант 3**

**Задание:** 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.

Код программы:

#include <iostream>  
#include <queue>  
#include <mutex>  
#include <thread>  
#include <chrono>  
#include <fstream>  
#include <vector>  
#include <algorithm>  
#include <windows.h>  
  
using namespace std;  
  
class Task {  
public:  
 string operation;  
 string fileName;  
  
 Task(string op, string name) : operation(op), fileName(name) {}  
};  
  
class TaskQueue {  
private:  
 queue<Task> tasks;  
 mutex mtx;  
public:  
 void addTask(const Task &task) {  
 lock\_guard<mutex> lock(mtx);  
 tasks.push(task);  
 }  
  
 Task getTask() {  
 lock\_guard<mutex> lock(mtx);  
 Task task = tasks.front();  
 tasks.pop();  
 return task;  
 }  
  
 bool empty() const {  
 return tasks.empty();  
 }  
};  
  
void createFile() {  
 ofstream file("filename.txt");  
 if (file.is\_open()) {  
 cout << "Файл создан." << endl;  
 } else {  
 cout << "Ошибка создания файла." << endl;  
 }  
 file.close();  
}  
  
void deleteFile() {  
 if (remove("filename.txt") == 0) {  
 cout << "Файл удален." << endl;  
 } else {  
 cout << "Ошибка удаления файла." << endl;  
 }  
}  
  
void renameFile() {  
 if (rename("filename.txt", "newfilename.txt") == 0) {  
 cout << "Файл переименован." << endl;  
 } else {  
 cout << "Ошибка переименования файла." << endl;  
 }  
}  
  
void displayFile() {  
 string line;  
 ifstream file("filename.txt");  
 if (file.is\_open()) {  
 while (getline(file, line)) {  
 cout << line << endl;  
 }  
 file.close();  
 } else {  
 cout << "Ошибка отображения файла." << endl;  
 }  
}  
  
void addRecordToFile() {  
 string record;  
 cout << "Введите запись для добавления в файл: ";  
 cin.ignore();  
 getline(cin, record);  
 ofstream file("filename.txt", ios::app);  
 if (file.is\_open()) {  
 file << record << endl;  
 cout << "Запись добавлена в файл." << endl;  
 } else {  
 cout << "Ошибка добавления записи в файл." << endl;  
 }  
 file.close();  
}  
  
void deleteRecordFromFile() {  
 string recordToDelete;  
 cout << "Введите запись для удаления из файла: ";  
 cin.ignore();  
 getline(cin, recordToDelete);  
 ifstream file("filename.txt");  
 if (file.is\_open()) {  
 string line;  
 string content = "";  
 while (getline(file, line)) {  
 if (line != recordToDelete) {  
 content += line + "\n";  
 }  
 }  
 file.close();  
 ofstream outputFile("filename.txt");  
 if (outputFile.is\_open()) {  
 outputFile << content;  
 cout << "Запись удалена из файла." << endl;  
 } else {  
 cout << "Ошибка удаления записи из файла." << endl;  
 }  
 } else {  
 cout << "Ошибка удаления записи из файла." << endl;  
 }  
}  
  
void executeTask(const int taskNum) {  
 switch (taskNum) {  
 case 1:  
 createFile();  
 break;  
 case 2:  
 deleteFile();  
 break;  
 case 3:  
 renameFile();  
 break;  
 case 4:  
 displayFile();  
 break;  
 case 5:  
 addRecordToFile();  
 break;  
 case 6:  
 deleteRecordFromFile();  
 break;  
 default:  
 cout << "Неверный номер задачи. Пожалуйста, введите действительный номер задачи от 1 до 6. " << endl;  
 }  
}  
  
void threadWorker(queue<pair<int, time\_t>> &tasksQueue, mutex &mtx) {  
 while (true) {  
 pair<int, time\_t> task;  
 {  
 lock\_guard<mutex> lock(mtx);  
 if (!tasksQueue.empty()) {  
 task = tasksQueue.front();  
 tasksQueue.pop();  
 } else {  
 break; // Выход из цикла, если очередь пуста  
 }  
 }  
 executeTask(task.first);  
 }  
}  
  
int main() {  
 SetConsoleCP(1251);  
 SetConsoleOutputCP(CP\_UTF8);  
 setlocale(LC\_ALL, "Russian");  
  
 queue<pair<int, time\_t>> tasksQueue;  
 mutex mtx;  
 vector<thread> threads;  
 int numThreads = 1;  
 thread worker(threadWorker, std::ref(tasksQueue), std::ref(mtx)); // передача аргументов по ссылке  
 threads.push\_back(move(worker));  
 int taskNum;  
 while (true) {  
 cout << "Введите номер действия (1: создать файл, 2: удалить файл, 3: переименовать файл, 4: отобразить файл, 5: добавить запись в файл, 6: удалить запись из файла, 0: выйти): ";  
 cin >> taskNum;  
 if (taskNum == 0) {  
 break; // Завершение цикла после ввода 0  
 }  
 executeTask(taskNum);  
 }  
 for (thread &t: threads) {  
 t.join();  
 }  
 return 0;  
}

Работа программы:

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Выводы: ходе данной лабораторной работы мы познакомились с ключевыми компонентами стандартной библиотеки шаблонов (STL) в C++, включая контейнеры (например, вектор, список), итераторы и алгоритмы.Мы изучили применение контейнеров для хранения и управления данными, используя различные классы контейнеров в STL в зависимости от требуемых операций и характеристик данных. Поняли, как использовать итераторы для обхода элементов контейнера, что позволяет нам применять алгоритмы к элементам контейнера без зависимости от его внутренней структуры. Освоили применение алгоритмов STL для выполнения различных операций над контейнерами, включая сортировку, поиск, фильтрацию и трансформацию данных.