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: Объектно-ориентированное программирование

тема: «**Потоки в С++**»

Выполнил: ст. группы ПВ-233

Ситников Алексей Павлович

Проверил:

Белгород 2025 г.

**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.

Вариант 3 (13)

Задание 1

Один поток удаляет пробелы в строке и вставляет их в случайное место, а другой поток выполняет циклический сдвиг текста. Произвести синхронный вывод при каждой итерации. Показать выполнение работы программы в синхронном и асинхронном режимах.  
  
Создано два класса:  
Изображение выглядит как текст, снимок экрана, линия, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Задача класса «MoveSpace» переставлять пробелы, для этого перегружен оператор функтор, а также для реализации алгоритма приватный метод «dataInVector», который определяет есть ли значение в векторе.

Задача класса «ShiftString» сдвигать строку, для этого перегружен оператор функтор и реализован адлгоритм.

Модуль заголовочных файлов:

#ifndef **UNTITLED10\_CLASSES\_H**#define **UNTITLED10\_CLASSES\_H**#include <vector>  
#include <cstdlib>  
#include <iostream>  
  
class MoveSpace{  
 int dataInVector(std::vector<int> &data, int &value, int &indx);  
public:  
 void operator**()**(char \*a);  
};  
  
  
class ShiftString{  
public:  
 void operator**()**(char \*a, int shift);  
};  
  
#endif //UNTITLED10\_CLASSES\_H

Модуль реализации методов:

#include "classes.h"  
#include <mutex>  
#include <thread>  
#include <chrono>  
  
  
std::mutex m;  
  
int MoveSpace::dataInVector(std::vector<int> &data, int &value, int &indx){  
 for(int i = 0; i < indx; i++){  
 if(data**[**i**]** == value){  
 return 1;  
 }  
 }  
 return 0;  
}  
  
void MoveSpace::operator**()**(char \*a){  
 std::lock\_guard<std::mutex> lock(m);  
 char \*begin = a;  
 std::vector<int> dataBefore;  
 int index = 0;  
 while (\*begin!='\0'){  
 if(\*begin == ' '){  
 dataBefore.push\_back(index);  
 }  
 begin++;  
 index++;  
 }  
 if(dataBefore.empty()){  
 return;  
 }  
 int countSpace = (int)dataBefore.size();  
 std::vector<int> dataAfter(countSpace);  
 for(int i = 0; i < countSpace; i++){  
  
 dataAfter**[**i**]** = std::rand() % index;  
 while (dataAfter**[**i**]** == dataBefore**[**i**]** || dataInVector(dataAfter, dataAfter**[**i**]**, i)){  
 dataAfter**[**i**]** = std::rand() % index;  
 }  
 }  
 int flag = 0;  
 for(int i = 0; i < index; i++){  
 if(dataInVector(dataAfter, i, index)){  
 if(a[i] == ' '){  
 continue;  
 }  
 if(flag == 0){  
 int index\_space = -1;  
 for(int j = i; j < index; j++){  
 if(a[j] == ' '){  
 index\_space = j;  
 break;  
 }  
 }  
 if(index\_space == -1){  
 return;  
 }  
 for(int j = index\_space; j > i; j--){  
 a[j] = a[j-1];  
 }  
 }  
 else{  
 int index\_space = -1;  
 for(int j = i; j >= 0; j--){  
 if(a[j] == ' '){  
 index\_space = j;  
 break;  
 }  
 }  
 if(index\_space == -1){  
 return;  
 }  
 for(int j = index\_space; j < i; j++){  
 a[j] = a[j+1];  
 }  
 }  
 a[i] = ' ';  
 }  
 else{  
 if(a[i] == ' '){  
 flag++;  
 }  
 }  
 std::this\_thread::sleep\_for(std::chrono::milliseconds(100));  
 }  
  
}  
  
  
  
void ShiftString::operator**()**(char \*a, int shift){  
 std::lock\_guard<std::mutex> lock(m);  
 char \*begin = a;  
 int size = 0;  
 while (\*begin != '\0'){  
 begin++;  
 size++;  
 }  
 int count = 0;  
 int i = 0;  
 char temp1 = '\0';  
 char temp2;  
 while (count < size){  
 if(temp1 == '\0'){  
 temp1 = a[(i+shift)%size];  
 a[(i+shift)%size] = a[i];  
 count++;  
 i = (i+shift)%size;  
 }  
 else{  
 temp2 = a[(i+shift)%size];  
 a[(i+shift)%size] = temp1;  
 count++;  
 i = (i+shift)%size;  
 temp1 = temp2;  
 }  
 std::this\_thread::sleep\_for(std::chrono::milliseconds(100));  
 }  
}

Модуль main:

#include <iostream>  
#include "classes.h"  
#include <ctime>  
#include <thread>  
#include <chrono>  
#include <windows.h>  
  
  
  
  
void printString(char \*a){  
  
 char \*b = a;  
 while (\*b!='\0'){  
 std::cout **<<** \*b **<<** std::flush;  
 char g = \*b;  
 b++;  
  
 }  
 std::cout **<<** "\n";  
  
}  
  
  
  
int main(){  
 SetConsoleOutputCP(**CP\_UTF8**);  
 std::srand(static\_cast<unsigned int>(std::time(**NULL**)));  
  
  
  
 MoveSpace moveSpace;  
 char d[] = "qw e rty";  
 ShiftString shiftString;  
  
 std::thread t1(moveSpace, d);  
 std::thread t2(shiftString, d, 5);  
 t1.join();  
 t2.join();  
  
 std::thread t3(printString, d);  
 t3.join();  
  
  
 return 0;  
}

Вывод программы:

синхронный  
Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

асинхронный

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Задание 2.

Один поток выполняет подсчет количества гласных букв в тексте, а другой вставляет или удаляет случайным образом гласную букву. Произвести синхронный вывод при каждой итерации. Показать выполнение работы программы в синхронном и асинхронном режимах.

Модуль заголовочных файлов:

#ifndef **UNTITLED10\_CLASSES\_H**#define **UNTITLED10\_CLASSES\_H**#include <vector>  
#include <cstdlib>  
#include <iostream>  
  
class CountLetters{  
public:  
 int operator**()**(char \*a);  
};  
  
  
class letters{  
public:  
 void operator**()**(char \*a);  
};  
  
#endif //UNTITLED10\_CLASSES\_H

Модуль реализации методов:

#include "classes.h"  
#include <mutex>  
  
std::mutex m;  
  
int CountLetters::operator**()**(char \*a){  
 std::lock\_guard<std::mutex> lock(m);  
 char \*b = a;  
 int count = 0;  
 char arr[] = "aeyuio";  
 while ( \*b!= '\0'){  
 for(int i = 0; i < 6; i++){  
 if(\*b == arr[i]){  
 count++;  
 break;  
 }  
 }  
 b++;  
 }  
 return count;  
}  
  
int valueInArray(char \*a, char v){  
 char \*begin = a;  
 while (\*begin!='\0'){  
 if(\*begin == v){  
 return 1;  
 }  
 begin++;  
 }  
 return 0;  
}  
  
void letters::operator**()**(char \*a){  
 std::lock\_guard<std::mutex> lock(m);  
 srand(static\_cast<unsigned int>(time(0)));  
  
 char \*begin = a;  
 int size = 0;  
 while (\*begin != '\0'){  
 size++;  
 begin++;  
 }  
 char arr[] = "aeyuio";  
 int t = rand() % 2;  
 if(t == 0){  
 int i = rand() % size;  
 while (!valueInArray(arr ,a[i])){  
 i = rand() % size;  
 }  
 a[i] = ' ';  
 }  
 else{  
 int i = rand() % size;  
 while (valueInArray(arr, a[i])){  
 i = rand() % size;  
 }  
 a[i] = arr[rand()%6];  
 }  
}

Модуль main:

#include <iostream>  
#include "classes.h"  
#include <windows.h>  
  
  
  
  
struct ThreadParams {  
 char\* str;  
 CountLetters\* countLetters;  
 letters\* shiftString;  
 DWORD\* result;  
};  
  
DWORD **WINAPI** CountLettersThread(LPVOID lpParam) {  
 ThreadParams\* params = static\_cast<ThreadParams\*>(lpParam);  
 \*params->result = (\*params->countLetters)**(**params->str**)**; // Сохраните результат как int  
 return 0;  
}  
  
  
DWORD **WINAPI** ShiftStringThread(LPVOID lpParam) {  
 ThreadParams\* params = static\_cast<ThreadParams\*>(lpParam);  
 (\*params->shiftString)**(**params->str**)**;  
  
 return 0;  
}  
  
typedef struct data{  
 char\* data1;  
 unsigned long data2;  
}date;  
  
DWORD **WINAPI** PrintThread(LPVOID lpParam) {  
 date ans = \*static\_cast<date\*>(lpParam);  
  
 std::cout **<<** ans.data1 **<<** " " **<<** ans.data2;  
 return 0;  
}  
  
  
  
int main() {  
  
  
 CountLetters C1;  
 letters C2;  
 char a[] = "wfouwdif";  
 DWORD resultCountLetters;  
 ThreadParams params1 = { a, &C1, nullptr, &resultCountLetters };  
 ThreadParams params2 = { a, nullptr, &C2 };  
  
 HANDLE hThread1 = CreateThread(  
 nullptr,  
 0,  
 ShiftStringThread,  
 &params2,  
 0,  
 nullptr  
 );  
  
 HANDLE hThread2 = CreateThread(  
 nullptr,  
 0,  
 CountLettersThread,  
 &params1,  
 0,  
 nullptr  
 );  
  
  
  
 WaitForSingleObject(hThread1, **INFINITE**);  
 WaitForSingleObject(hThread2, **INFINITE**);  
 CloseHandle(hThread1);  
 CloseHandle(hThread2);  
  
 date ans;  
 ans.data2 = resultCountLetters;  
 ans.data1 = a;  
  
  
 HANDLE hPrintThread1 = CreateThread(  
 nullptr, // Дефолтные атрибуты безопасности  
 0, // Дефолтный размер стека  
 PrintThread, // Указатель на функцию потока  
 &ans, // Указатель на параметры  
 0, // Поток будет запущен сразу  
 nullptr // Идентификатор потока не нужен  
 );  
  
 WaitForSingleObject(hPrintThread1, **INFINITE**);  
 CloseHandle(hPrintThread1);  
  
  
  
 return 0;  
}

Вывод программы:

асинхронный   
Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

синхронный

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

**Вывод:** в ходе проделанной работы я изучил потоки и как с ними работать.