БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Кафедра информационных технологий автоматизированных систем

Факультет ФИТУ

Специальность АСОИ

Контрольная работа по модулю 3

по дисциплине «Системное программное обеспечение», часть 1

ВАРИАНТ 1

Выполнил студент: Гриб В.А.

группа 820602

Зачетная книжка № 82060030

Минск 2020

**Задание**

Использовать выполненную КР по модулю 2.

Для всех вариантов необходимо выполнить следующее:

•  используемые функции оформить в виде библиотеки DLL;

•  создать новый проект для демонстрации работы с DLL и функциями;

•  в **нечетных** номерах заданий необходимо использовать динамическое связывание без импорта, в **четных** – статическое связывание с импортом;

•  при динамическом связывании необходимо в проекте с главной программой объявить указатели на функции, содержащиеся в DLL;

•  при статическом связывании необходимо создать специальный заголовочный файл с объявлениями функций;

**Решение**

Мой вариант имеет нечетное значение, поэтому я использую динамическое связывание без импорта

**Код программы:**

**Файл “Dates.cpp”**

#include <stdio.h>

#include <thread>

#include <iostream>

#include <fstream>

#include <mutex>

#include <condition\_variable>

**static** **const** **int** MAX\_COUNT = 5;

**struct** Date {

**int** year;

**int** month;

**int** day;

};

**struct** Dates {

**int** count = 0; //количество имеющихся элементов в массиве

Date dates[100];

};

mutex kLock;

condition\_variable kCv;

**bool** kReady = **false**;

**bool** kProcessed = **false**;

**void** secondThread(Dates \*dates){

ofstream datesFile("dates.txt");

**int** currentCount = dates->count;

**while** (currentCount < MAX\_COUNT) {

// дождаться передачи управления от главного потока

{

unique\_lock<std::mutex> lk(kLock);

kCv.wait(lk, [] { **return** kReady; } );

}

**if** (dates->count != currentCount) {

currentCount = dates->count;

Date date = dates->dates[currentCount-1];

datesFile << "date[" << currentCount << "]: { day: " << date.day << ", month: " << date.month << ", year: " << date.year << " } \n" << endl;

}

{

std::lock\_guard<std::mutex> lk(kLock);

kProcessed = **true**;

kCv.notify\_one();

}

}

datesFile.close();

}

\_\_declspec(dllexport) **void** createdAndWriteToFileDates(Dates \*dates) {

thread secondThr(secondThread, dates);

secondThr.detach();

**while** (dates->count < MAX\_COUNT) {

Date tmp;

cout << "Enter year: ";

cin >> tmp.year;

cout << endl << "Enter month: ";

cin >> tmp.month;

cout << endl << "Enter day: ";

cin >> tmp.day;

cout << endl;

dates->dates[dates->count] = tmp;

dates->count++;

{

// передать управление второму потоку

std::lock\_guard<std::mutex> lk(kLock);

kReady = **true**;

kCv.notify\_one();

}

// дождаться выполнения второго потока

std::unique\_lock<std::mutex> lk(kLock);

kCv.wait(lk, []{**return** kProcessed;});

}

}

**Файл “main.cpp”**

#include <iostream>

#include <windows.h>

**using** **namespace** std;

**struct** Date {

**int** year;

**int** month;

**int** day;

};

**struct** Dates {

**int** count = 0; //количество имеющихся элементов в массиве

Date dates[100];

};

**typedef** **void**(\_\_cdecl \* CREATE\_AND\_WRITE\_TO\_FILE\_DATES)(Dates \*dates);

**int** main(**int** argc, **const** **char** \* argv[]) {

HANDLE hLibrary;

CREATE\_AND\_WRITE\_TO\_FILE\_DATES createAndWriteToFileDates;

Dates dates = {0};

hLibrary = LoadLibrary("MyDate.dll");

**if** (!hLibrary) **return** 0;

createAndWriteToFileDates = (CREATE\_AND\_WRITE\_TO\_FILE\_DATES)GetProcAddress(hLibrary,"createdAndWriteToFileDates");

**if** (createAndWriteToFileDates) createAndWriteToFileDates(&dates);

FreeLibrary(hLibrary);

**return** 0;

}

**Выводы:**

В ходе выполнения данной работы мною были изучены средства для создания библиотек DLL путём динамического связывания в C++;