Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Кафедра информационных технологий автоматизированных систем

Факультет ИТиУ
Специальность АСОИ

Индивидуальная практическая работа по модулю 3 по дисциплине «Системное программное обеспечение», часть 1 «Динамически подключаемые библиотеки» Вариант №1

Выполнил:
Ст. Гр. 820601
Шведов А.Р
Зачетная книжка No 82060145

Минск 2020

1. 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;

2. Ход работы

2.1.Листинг программы

В программе используется динамическое связывание без импорта

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,"createdAn
dWriteToFileDates");

    if (createAndWriteToFileDates) createAndWriteToFileDates(&dates);

    FreeLibrary(hLibrary);

    return 0;
}
```

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];
};

int enter_int(){
    int n;

    while(true){
```

```
        cin >> n;
        if (!cin){  // == cin.fail()
            cout << "error. Try again\n";
            cin.clear();
            while(cin.get()!='\n');
        }else break;
    }
    return n;
}

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: ";
        tmp.year = enter_int();
        cout << endl << "Enter month: ";
        tmp.month = enter_int();
        cout << endl << "Enter day: ";
        tmp.day = enter_int();
        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;});
    }
}
```

3. Выводы

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