Міністерство освіти і науки України
Прикарпатський національний університет
імені Василя Стефаника

О. В. Махней

Лабораторний практикум з системного програмування

Методичні рекомендації
до проведення лабораторних занять

Івано-Франківськ
2022

УДК 004.45
ББК 32.973.1
М36

Рекомендовано до друку Вченою радою факультету математики та інформатики Прикарпатського національного університету імені Василя Стефаника (протокол № 10 від 10 червня 2022 р.).

Рецензенти:

Мазуренко В. В., кандидат фізико-математичних наук, доцент (Прикарпатський національний університет імені Василя Стефаника),

Горєлов В. О., кандидат технічних наук, доцент (Прикарпатський національний університет імені Василя Стефаника)

М36 Махней О. В. Лабораторний практикум з системного програмування : методичні рекомендації до проведення лабораторних занять. Івано-Франківськ : Голіней, 2022. 28 с.

Наведено методичні рекомендації до виконання лабораторних робіт з системного програмування. Призначено для проведення лабораторних занять з курсів «Системне програмування», «Системне програмування та спеціалізовані мови програмування».

Для студентів спеціальності «прикладна математика». Може бути корисним для студентів галузей знань «математика і статистика», «інформаційні технології».

Зміст

Передмова

Лабораторний практикум з системного програмування містить методичні рекомендації і завдання до лабораторних робіт, присвячені розробці системного програмного забезпечення для операційної системи Windows. Практикум орієнтований на використання мови програмування C++ для вивчення навчальної дисципліни «Системне програмування».

До складу практикуму входять вісім лабораторних робіт, сім з яких передбачають використання засобів Windows API, тобто інтерфейсу доступу до функцій операційної системи Windows, для роботи з файлами, каталогами, процесами і потоками.

Кожна лабораторна робота містить короткий опис звертання до функцій Win32 API і деяких функцій з бібліотек C++, потрібних для виконання лабораторної роботи, та завдання для виконання лабораторної роботи. Наявність значної кількості варіантів дозволяє стимулювати самостійну роботу студентів.

Лабораторна робота № 1. Читання і запис файлів засобами C++

Стандартні засоби C++ для читання і запису файлів

Потрібна директива #include <fstream>. Для створення потоків введення з файлу, виведення у файл і введення та виведення використовують класи ifstream, ofstream і fstream відповідно. Ім'я файлу задається в конструкторі або використанням методу open(filename). Для запису текстових файлів використовують оператор <<, а бінарних — метод write(buf,n), де buf — буфер (масив), n — кількість байт для запису. Для читання використовують оператор >>, методи get(buf,n), getline(buf,n), read(buf,n) (читається $n-1$ символ). Для закриття файлу використовують метод close().

При читанні для зміни позиції у файлі використовують метод seekg(n,m), де n — нова позиція для читання у файлі або її зміщення, а m — спосіб позиціонування:

ios_base::beg — відлік позиції від початку файлу (абсолютне позиціонування),

ios_base::end — відлік позиції від кінця файлу,

ios_base::cur — зміщення від поточної позиції (за замовчуванням).

Першому байту у файлі відповідає позиція 0. При запису для зміни позиції у файлі використовують метод seekp(n,m) з аналогічними аргументами.

Для роботи з рядками зручно використовувати функції strcpy(d,s) (копіює рядок s в d) і strlen(s) (визначає довжину рядка s) з бібліотеки cstring.

Завдання

1. Написати програму surname_lab1_1, яка запише у файл з назвою surname_lab1.txt (surname — прізвище студента) наступний текст: «Студент/студентка групи ПМ-41 Прізвище Ім'я хоче вчитись добре», де підставляється прізвище і ім'я студента.

2. Написати програму surname_lab1_2, яка:

• прочитає з $(3+n)$-ї позиції файлу surname_lab1.txt 20 байтів, виведе їх на екран і в файл surname_lab1w.txt (n — номер студента у списку групи);

• у файлі surname_lab1.txt поміняє місцями ім'я і прізвище (при цьому має перезаписуватись лише частина тексту з прізвищем і іменем).

Лабораторна робота № 2. Читання і запис файлів засобами Windows API

Засоби Windows API для читання і запису файлів

Для роботи з функціями Windows API в C++ потрібна директива #include <windows.h>.

Для відкриття (створення) файлу використовується функція

```
HANDLE CreateFile (
  LPCTSTR FileName,  // ім'я файлу
  DWORD Access,  // тип доступу до файлу
  DWORD Sharing,  // тип спільного доступу
  LPSECURITY_ATTRIBUTES Attrib,  // вказівник на
  // структуру атрибутів безпеки
  DWORD CreationDistr,  // що робити, коли файл існує
  DWORD Flags,  // атрибути і прапорці файлу
  HANDLE Template)  // дескриптор файлу, атрибути якого
  // треба скопіювати
```

Функція повертає дескриптор відкритого файлу або INVALID_HANDLE_VALUE в разі помилки. Опис всіх можливих значень аргументів є в документації по Windows API (є в інтернеті). Типовий виклик для відкриття існуючого файлу:

```
HANDLE f1=CreateFile("name.dat", FILE_ALL_ACCESS, 0,
  NULL, OPEN_ALWAYS, FILE_ATTRIBUTE_NORMAL,
  NULL);
```

Для створення нового файлу краще використовувати значення CREATE_ALWAYS замість OPEN_ALWAYS.

За читання з файлу і запис у файл відповідають дві наступні функції:

```
BOOL ReadFile (
  HANDLE File,  // дескриптор файлу
  LPVOID Buf,  // адреса буфера, в який читаються дані
  DWORD BytesToRead,  // скільки байтів прочитати
  LPDWORD BytesRead,  // адреса змінної, що міститиме
  // кількість прочитаних байтів
  LPOVERLAPPED Overlap)  // адреса структури перекриття
BOOL WriteFile (
  HANDLE File,  // дескриптор файлу
  LPVOID Buf,  // адреса буфера, в якому зберігаються дані
  // для запису
  DWORD BytesToWrite,  // скільки байтів записати
  LPDWORD BytesWriten,  // адреса змінної, що міститиме
  // кількість записаних байтів
  LPOVERLAPPED Overlap)  // адреса структури перекриття
```

Якщо функція ReadFile або WriteFile повертає FALSE, то має місце помилка введення або виведення. За відсутності структури перекриття останнім аргументом цих функцій ставиться NULL.

При читанні або записі для зміни позиції у файлі використовують функцію

```
DWORD SetFilePointer(
  HANDLE File,  // дескриптор відкритого файлу
  LONG BytesToMove,  // нова позиція для читання чи запису
  // у файлі або її зміщення
  PLONG DistanceToMoveHigh,  // адреса старших 32 бітів
  // нової позиції читання/запису
  DWORD dwMoveMethod)  // режим зміни позиції у файлі
```

Функція SetFilePointer повертає молодші 32 біти нової позиції у файлі або INVALID_SET_FILE_POINTER у випадку помилки. Першому байту у файлі відповідає позиція 0. Параметр dwMoveMethod може набувати значень: FILE_BEGIN — відлік позиції від початку файлу (абсолютне позиціонування), FILE_END — відлік позиції від кінця файлу,

FILE_CURRENT — зміщення від поточної позиції.

Закриває файл функція

```
BOOL CloseHandle(
  HANDLE File)  // дескриптор файлу
```

Визначити розмір файлу можна, зокрема, з допомогою функції

```
DWORD GetFileSize(
  HANDLE File,  // дескриптор файлу
  LPDWORD lpFileSizeHigh)  // адреса старших 32 бітів
  // довжини файлу
```

Результатом цієї функції є молодші 32 біти довжини файлу.

Завдання

1. З допомогою функцій Windows API написати програму surname_lab2_1, яка запише у файл з назвою surname_lab2.txt (surname — прізвище студента) наступний текст: «Студент/студентка групи ПМ-41 Прізвище Ім'я хоче вчитись добре», де підставляється прізвище і ім'я студента.

2. З допомогою функцій Windows API написати програму surname_lab2_2, яка:

- прочитає з $(3+n)$-ї позиції файлу surname_lab2.txt 20 байтів, виведе їх на екран і в файл surname_lab2w.txt (n — номер студента у списку групи);
- у файлі surname_lab2.txt поміняє місцями ім'я і прізвище (при цьому має перезаписуватись лише частина тексту з прізвищем і ім'ям);
- виведе на екран розмір файлу surname_lab2.txt у байтах.

Лабораторна робота № 3. Керування файлами і каталогами засобами Windows API

Засоби Windows API для керування файлами

Для роботи з функціями Windows API в C++ потрібна директива #include <windows.h>.

Новий каталог створюється функцією

```
BOOL CreateDirectory (
  LPCTSTR PathName,  // ім'я каталогу
  LPSECURITY_ATTRIBUTES lpSecurityAttributes)
  // адреса атрибутів безпеки
```

Якщо атрибутів безпеки немає, то другим аргументом має бути NULL. Якщо каталог успішно створено, то функція поверне ненульове значення.

Копіювати файли найлегше з допомогою функції

```
BOOL CopyFile(
  LPCTSTR ExistingFileName,  // ім'я файлу-джерела
  LPCTSTR NewFileName,  // ім'я файлу-одержувача
  BOOL bFailIfExists)  // дії, якщо файл вже існує
```

Якщо файл вже існує і третім аргументом є значення TRUE, то функція повертає нульове значення. Якщо файл вже існує і третім аргументом є значення FALSE, то файл перезаписується.

Аналогічно здійснюється перейменування і видалення файлів:

```
BOOL MoveFile(
  LPCTSTR ExistingFileName,  // ім'я файлу-джерела
  LPCTSTR NewFileName)  // ім'я перейменованого файлу
BOOL DeleteFile(
  LPCTSTR ExistingFileName)  // ім'я файлу, що видаляється
```

Функція DeleteFile не зберігає видалені файли в кошику, будьте уважні і обережні!

Функція

```
BOOL SetFileAttributes(
  LPCTSTR lpFileName,  // ім'я файлу
  DWORD dwFileAttributes)  // атрибути
```

присвоює файлу вказані атрибути. Найчастіше використовують атрибути: FILE_ATTRIBUTE_HIDDEN — схований файл, FILE_ATTRIBUTE_NORMAL — немає інших атрибутів, FILE_ATTRIBUTE_READONLY — файл лише для читання, FILE_ATTRIBUTE_SYSTEM — системний файл.

Функція

```
BOOL LockFile(
  HANDLE hFile,  // дескриптор файлу
  DWORD dwFileOffsetLow,  // молодше слово зміщення
  DWORD dwFileOffsetHigh,  // старше слово зміщення
  DWORD nNumberOfBytesToLockLow,  // молодше слово
  // довжини
  DWORD nNumberOfBytesToLockHigh)  // старше слово
  // довжини
```

блокує частину файлу заданої в байтах довжини, починаючи від зміщення. Іншим процесам буде відмовлено в доступі навіть для читання. Файл має бути відкритий функцією CreateFile.

Функція UnLockFile з аналогічними аргументами розблоковує заблокований файл.

Для пошуку файлів у каталозі використовують дві функції. Перша починає пошук і повертає дескриптор пошуку:

```
HANDLE FindFirstFile (
  LPCTSTR FileName,  // які файли шукати, можна вказувати
  // маску символами * і ?
  LPWIN32_FIND_DATA FindData)  // вказівник на
  // структуру з отриманою інформацією
```

Для продовження пошуку використовується функція:

```
BOOL FindNextFile (
  HANDLE FindFile,  // дескриптор пошуку
  LPWIN32_FIND_DATA FindData)  // вказівник на
  // структуру з отриманою інформацією
```

Функція FindClose завершує пошук файлу. Вона описана так:

```
BOOL FindClose (
  HANDLE FindFile)  // дескриптор пошуку
```

Приклад пошуку у C++:

```
WIN32_FIND_DATA FindData;
HANDLE MyFile;
MyFile = FindFirstFile ("d:\\*.*", &FindData);
if (MyFile != INVALID_HANDLE_VALUE)
  do  cout<< FindData.cFileName<<endl;
  while (FindNextFile (MyFile, &FindData));
```

FindClose (MyFile);

Для перейменування і видалення файлів зручно користуватись стандартними функціями C++ для роботи з рядками strlen, strcpy, strcat, strcmp та іншими.

Функція GetLastError() повертає код помилки, її можна використовувати для обробки помилок введення/виведення.

Завдання

1. Створити текстовий файл з назвою surname_lab3.txt (surname — прізвище студента), який міститиме текст «Цю лабораторну роботу виконав студент групи ПМ-41 Ім'я Прізвище. Номер студента у списку групи #.». Замість слів «Ім'я Прізвище» підставляється ім'я і прізвище студента, замість «#» підставляється номер студента у списку групи. Це завдання можна виконати з допомогою програми Блокнот.

2. З допомогою функцій Windows API написати програму surname_lab3_1, яка:

- створить каталог з назвою «lab3dir»;
- розмістить у створеному каталозі дві копії файлу surname_lab3.txt з тим самим ім'ям і з ім'ям «lab3.dat»;
- розмістить у створеному каталозі дві копії файлу surname_lab3.txt з іменами «1xx.txt» і «2xx.dat», де «xx» — номер студента у списку групи;
- для файлу «2xx.dat» задасть атрибути системного схованого файлу з дозволом лише для читання;
- заблокує 100 перших байтів оригінального файлу surname_lab3.txt для інших процесів (монопольний доступ);
- очікуватиме введення символу з клавіатури для того щоб можна було переконатись в заблокованості файлу;
- розблокує заблокований файл.

3. Переконатись в заблокованості файлу, переглянути копії файлів.

4. З допомогою функцій Windows API написати програму surname_lab3_2, яка:

- виведе на екран вміст каталогу lab3dir (з допомогою функцій FindFirstFile і FindNextFile);

• всі файли каталогу lab3dir з розширенням «.dat» перейменує на файли з розширенням «.txt»;

• знищить файл lab3.txt з каталогу lab3dir.

Лабораторна робота № 4. Редагування файлів з допомогою їхнього відображення в пам'ять

Засоби Windows API для відображення файлів

Для роботи з функціями Windows API в C++ потрібна директива #include <windows.h>.

Для відкриття файлу використовується функція

```
HANDLE CreateFile (
  LPCTSTR FileName,  // ім'я файлу
  DWORD Access,  // тип доступу до файлу
  DWORD Sharing,  // тип спільного доступу
  LPSECURITY_ATTRIBUTES Attrib,  // вказівник на
  // структуру атрибутів безпеки
  DWORD CreationDistr,  // що робити, коли файл існує
  DWORD Flags,  // атрибути і прапорці файлу
  HANDLE Template)  // дескриптор файлу, атрибути якого
  // треба скопіювати
```

Для відкриття існуючого файлу CreationDistr може бути рівним OPEN_EXISTING. Для редагування файлу Access має бути рівним FILE_ALL_ACCESS. Результатом функції є дескриптор файлу або INVALID_HANDLE_VALUE.

Визначити розмір файлу можна, зокрема, з допомогою функції

```
DWORD GetFileSize(
  HANDLE File,  // дескриптор файлу
  LPDWORD lpFileSizeHigh)  // адреса старших 32 бітів
  // довжини файлу
```

Для створення об'єкта відображення файлу використовується функція

```
HANDLE CreateFileMapping(
```

```
  HANDLE hFile,  // дескриптор відкритого файлу
  LPSECURITY_ATTRIBUTES lpsa,  // атрибути захисту
  // об'єкта відображення
  DWORD dwProtect,  // можливості доступу до файлу при
  // його відображенні
  DWORD dwMaximumSizeHigh,  // старша частина значення
  // максимального розміру об'єкта відображення файлу
  DWORD dwMaximumSizeLow,  // молодша частина значення
  // максимального розміру об'єкта відображення файлу
  LPCTSTR lpMapName)  // вказівник на рядок з ім'ям
  // об'єкта відображення
```

Результатом функції є дескриптор об'єкта відображення файлу.

Для виділення пам'яті і відображення туди файлу використовується функція

```
LPCVOID MapViewOfFile(
  HANDLE hMapObject,  // дескриптор об'єкта відображення
  DWORD dwAccess,  // тип доступу до об'єкта відображення
  DWORD dwOffsetHigh,  // старша частина зсуву початку
  // відображуваної ділянки в файлі
  DWORD dwOffsetLow,  // молодша частина зсуву початку
  // відображуваної ділянки в файлі
  SIZE_T cbMap)  // розмір відображуваної ділянки файлу
```

Результатом функції є вказівник на область відображення файлу в пам'яті.

Для звільнення пам'яті після відображення файлу використовується функція

```
BOOL UnmapViewOfFile(
  LPCVOID lpBaseAddress) // вказівник на область
  // відображення
```

Для знищення об'єкта відображення або закриття файлу використовується функція

```
BOOL CloseHandle(HANDLE hMapObject)  // дескриптор
  // об'єкта відображення (файлу)
```

Завдання

З допомогою функцій Windows API написати програму surname_lab4 (surname — прізвище студента), яка:

- прочитає з консолі ім'я файлу;
- відобразить файл у пам'ять;
- зашифрує його з використанням відображення в пам'ять методом простої перестановки байтів з періодом 5 і ключем, відповідним номеру студента у списку групи (ключ береться з таблиці).

Якщо файл з вказаним ім'ям не існує, то програма має вивести відповідне повідомлення. Гарантується, що розмір файлу не перевищуватиме 1 Гбайт. Якщо розмір файлу не є кратним 5 байтам, то останні один, два, три або чотири байти не переставляються. Тестувати програму потрібно на копії якогось файлу, інакше доведеться писати програму для його розшифровки.

Варіант	Ключ	Варіант	Ключ	Варіант	Ключ
1	35421	10	35142	19	43152
2	51243	11	25431	20	32154
3	41352	12	51432	21	43512
4	25143	13	42153	22	53421
5	32514	14	34521	23	21453
6	14253	15	45123	24	51423
7	24153	16	21534	25	31452
8	31254	17	15423	26	23541
9	51423	18	54231	27	45132

Наприклад, ключ 32514 означає, що в кожній групі з п'яти байтів на першому місці стоятиме третій байт, на другому — другий, на третьому — п'ятий, на четвертому — перший, на п'ятому — четвертий.

Лабораторна робота № 5. Керування процесами

Засоби Windows API для керування процесами

Для роботи з функціями Windows API в C++ потрібна директива #include <windows.h>.

Для створення процесу використовується функція

```
BOOL CreateProcess (
  LPCTSTR lpApplicationName,  // ім'я виконуваного файлу
  LPTSTR lpCommandLine,  // параметри командного рядка
  LPSECURITY_ATTRIBUTES lpsaProcess,  // вказівник на
  // атрибути захисту процесу
  LPSECURITY_ATTRIBUTES lpsaThread,  // вказівник на
  // атрибути захисту потоку
  BOOL bInheritHandles,  // дозвіл на спадкування
  // дескрипторів
  DWORD dwCreationFlags,  // параметри процесу
  LPVOID lpEnvironment,  // вказівник на змінні оточення
  LPCTSTR lpCurDir,  // поточний каталог
  LPSTARTUPINFO lpStartupInfo,  // вказівник на початкову
  // інформацію для процесу
  LPPROCESS_INFORMATION lpProcInfo)  // вказівник на
  // структуру з дескрипторами
```

Для очікування завершення створеного процесу використовується функція

```
DWORD WaitForSingleObject (
  HANDLE hObject,  // дескриптор процесу
  DWORD dwMilliseconds)  // тривалість очікування
```

Для закриття дескриптора використовується функція

```
BOOL CloseHandle(HANDLE hProcess) // дескриптор
```

Завдання

1. З допомогою функцій Windows API написати програму surname_lab5_1 (surname — прізвище студента), яка прочитає вхідні дані з консолі, запустить програму surname_lab5_2, передасть їй вхідні дані як елементи командного рядка і доче-

кається завершення запущеної програми.

2. Написати програму surname_lab5_2, яка прочитає передані їй аргументи командного рядка, виконає дії відповідно до заданого варіанту і виведе результати на екран.

Варіант	Задача
1	Знайти суму непарних чисел від числа L1 до числа L2.
2	Обчислити число Фібоначчі за номером n і формулою $F(n) = F(n-1) + F(n-2)$, $F(1) = 1$, $F(2) = 1$.
3	Якщо два рядка містять цілі числа зі знаком, то виводиться їхня сума, інакше — об'єднання рядків.
4	Рядок з одиниць і нулів записується у протилежному порядку, виводиться отриманий рядок і відповідне йому десяткове число.
5	Знайти суму квадратів цілих чисел від числа L1 до числа L2.
6	Рядок містить лише великі і малі латинські букви. Вивести рядок, отриманий з наявного зміною регістру букв, а також кількість букв у рядку.
7	Якщо натуральне число є степенем двійки, то виводиться показник степеня двійки, інакше виводиться відповідь «число не є степенем двійки».
8	Знайти суму кубів парних чисел від числа L1 до числа L2.
9	У рядку з одиниць і нулів одиниці замінюються на нулі, а нулі на одиниці, виводиться отриманий рядок і відповідне йому десяткове число.
10	Рядок не містить пропусків. Вивести рядок, записаний у протилежному порядку, а також кількість букв у рядку.

Лабораторна робота № 6. Створення багатопотокових програм в операційній системі Windows

Засоби Windows API для керування потоками

Для роботи з функціями Windows API в C++ потрібна директива #include <windows.h>.

Для створення потоку використовується функція

```
BOOL CreateThread (
  LPSECURITY_ATTRIBUTES lpsa,  // вказівник на
  // атрибути захисту потоку
  SIZE_T dwStackSize,  // розмір стека потоку в байтах
  LPTHREAD_START_ROUTINE lpStartAddr,  // вказівник
  // на функцію потоку
  LPVOID lpThreadParm,  // вказівник на аргумент для нового
  // потоку
  DWORD dwCreationFlags,  // параметри потоку
  LPDWORD lpThreadId)  // вказівник на ідентифікатор
  // потоку
```

Для очікування завершення створеного потоку використовується функція

```
DWORD WaitForSingleObject (
  HANDLE hObject,  // дескриптор потоку
  DWORD dwMilliseconds)  // тривалість очікування
```

Для закриття дескриптора використовується функція

```
BOOL CloseHandle(HANDLE hProcess)  // дескриптор
```

Завдання

З допомогою функцій Windows API написати програму surname_lab6 (surname — прізвище студента), яка прочитає вхідні дані з консолі, запустить окремий потік, передасть йому вхідні дані, дочекається завершення запущеного потоку і виведе результати на екран. Потік має виконати дії відповідно до заданого варіанту над переданими йому вхідними даними і передати результати батьківському потоку.

Варіант	Задача
1	Знайти суму непарних чисел від числа L1 до числа L2.
2	Обчислити число Фібоначчі за номером n і формулою $F(n) = F(n-1) + F(n-2)$, $F(1) = 1$, $F(2) = 1$.
3	Якщо два рядка містять цілі числа зі знаком, то виводиться їхня сума, інакше — об'єднання рядків.
4	Рядок з одиниць і нулів записується у протилежному порядку, виводиться отриманий рядок і відповідне йому десяткове число.
5	Знайти суму квадратів цілих чисел від числа L1 до числа L2.
6	Рядок містить лише великі і малі латинські букви. Вивести рядок, отриманий з наявного зміною регістру букв, а також кількість букв у рядку.
7	Якщо натуральне число є степенем двійки, то виводиться показник степеня двійки, інакше виводиться відповідь «число не є степенем двійки».
8	Знайти суму кубів парних чисел від числа L1 до числа L2.
9	У рядку з одиниць і нулів одиниці замінюються на нулі, а нулі на одиниці, виводиться отриманий рядок і відповідне йому десяткове число.
10	Рядок не містить пропусків. Вивести рядок, записаний у протилежному порядку, а також кількість букв у рядку.

Лабораторна робота № 7. Керування пріоритетами потоків

Засоби Windows API для керування потоками

Для роботи з функціями Windows API в C++ потрібна директива #include <windows.h>.

Для створення потоку використовується функція

```
BOOL CreateThread (
  LPSECURITY_ATTRIBUTES lpsa,  // вказівник на
  // атрибути захисту потоку
  SIZE_T dwStackSize,  // розмір стека потоку в байтах
  LPTHREAD_START_ROUTINE lpStartAddr,  // вказівник
  // на функцію потоку
  LPVOID lpThreadParm,  // вказівник на аргумент для нового
  // потоку
  DWORD dwCreationFlags,  // параметри потоку
  LPDWORD lpThreadId)  // вказівник на ідентифікатор
  // потоку
```

Для зміни пріоритету потік створюють призупиненим (з параметром CREATE_SUSPENDED).

Для очікування завершення створеного потоку використовується функція

```
DWORD WaitForSingleObject (
  HANDLE hObject,  // дескриптор потоку
  DWORD dwMilliseconds)  // тривалість очікування
```

Для закриття дескриптора використовується функція

```
BOOL CloseHandle(HANDLE hProcess)  // дескриптор
```

Для зміни пріоритету потоку використовують функцію

```
BOOL SetThreadPriority (
  HANDLE hThread,  // дескриптор потоку
  int nPriority)  // пріоритет потоку
```

Для відновлення роботи призупиненого потоку призначена функція

```
DWORD ResumeThread (
  HANDLE hThread)  // дескриптор потоку
```

Для визначення часу роботи потоку призначена функція

```
BOOL GetThreadTimes (
  HANDLE Thread,  // дескриптор потоку
  PFILETIME Created,  // вказівник на час в сотнях нс від
  // 01.01.1601 до створення потоку
  PFILETIME Exited,  // вказівник на час в сотнях нс від
  // 01.01.1601 до завершення потоку
  PFILETIME Kernel,  // вказівник на час в сотнях нс,
  // витрачений потоком на код операційної системи
  PFILETIME User)  // вказівник на час в сотнях нс,
  // витрачений потоком на код програми
```

Структура FILETIME оголошена так:

```
typedef struct FILETIME
  DWORD dwLowDateTime;  // молодша частина часу
  DWORD dwHighDateTime;  // старша частина часу
FILETIME, *PFILETIME, *LPFILETIME
```

Завдання

З допомогою функцій Windows API написати програму surname_lab7 (surname — прізвище студента), яка прочитає вхідні дані з консолі, запустить паралельно два потоки з заданими пріоритетами, передасть їм вхідні дані, дочекається завершення запущених потоків і виведе результати та тривалість виконання потоків на екран. Потоки мають виконати дії відповідно до заданого варіанту над переданими їм вхідними даними і передати результати батьківському потоку.

Варіант	Задача
1	Знайти суму парних чисел від числа L1 до числа L2. Пріоритет першого потоку — THREAD_PRIORITY_ABOVE_NORMAL, пріоритет другого потоку — THREAD_PRIORITY_IDLE.

Варіант	Задача
2	Обчислити число Фібоначчі за номером n і формулою $F(n) = F(n-1) + F(n-2)$, $F(1) = 1$, $F(2) = 1$. Пріоритет першого потоку — THREAD_PRIORITY_NORMAL, пріоритет другого потоку — THREAD_PRIORITY_LOWEST.
3	Знайти суму квадратів непарних чисел від числа L1 до числа L2. Пріоритет першого потоку — THREAD_PRIORITY_HIGHEST, пріоритет другого потоку — THREAD_PRIORITY_BELOW_NORMAL.
4	Знайти суму кубів непарних чисел від числа L1 до числа L2. Пріоритет першого потоку — THREAD_PRIORITY_ABOVE_NORMAL, пріоритет другого потоку — THREAD_PRIORITY_IDLE.
5	Обчислити число за номером n і формулою $F(n) = 2F(n-1) + F(n-3)$, $F(1) = 1$, $F(2) = 1$, $F(3) = 1$. Пріоритет першого потоку — THREAD_PRIORITY_HIGHEST, пріоритет другого потоку — THREAD_PRIORITY_BELOW_NORMAL.

Лабораторна робота № 8. Синхронізація потоків

Засоби Windows API для синхронізації потоків

Функції взаємоблокування

LONG InterlockedIncrement (PLONG Addend)

LONG InterlockedDecrement (PLONG Addend)

збільшують або зменшують на одиницю значення цілої змінної, на яку вказує вказівник Addend, і повертають отримане значення.

Для використання критичної секції спочатку потрібно оголосити глобальну змінну типу CRITICAL_SECTION і ініціалізувати її викликом функції

```
VOID InitializeCriticalSection (
  LPCRITICAL_SECTION lpCriticalSection)
  // вказівник на змінну
```

Функція

```
VOID EnterCriticalSection (
  LPCRITICAL_SECTION lpCriticalSection)
  // вказівник на змінну
```

блокує потік, який намагається виконати код критичної секції, якщо на цій критичній секції присутній інший потік. Потік, що очікує, розблокується після того, як інший потік виконає функцію

```
VOID LeaveCriticalSection (
  LPCRITICAL_SECTION lpCriticalSection)
  // вказівник на змінну
```

Функція

```
VOID DeleteCriticalSection (
  LPCRITICAL_SECTION lpCriticalSection)
  // вказівник на змінну
```

видаляє критичну секцію.

Для створення м'ютекса використовується функція

```
HANDLE CreateMutex (
  LPSECURITY_ATTRIBUTES lpsa,
  // вказівник на структуру атрибутів захисту
  BOOL bInitialOwner,  // true, якщо потік-створювач зразу
  // займає м'ютекс
  LPCTSTR lpMutexName)  // вказівник на ім'я м'ютекса
```

Для захоплення м'ютекса призначена функція WaitForSingleObject(...). Функція

```
BOOL ReleaseMutex (
  HANDLE hMutex)  // дескриптор м'ютекса
```

звільняє м'ютекс, яким володіє потік, що викликає функцію.

Семафор створюється функцією

```
HANDLE CreateSemaphore (
  LPSECURITY_ATTRIBUTES lpsa,
  // вказівник на структуру атрибутів захисту
  LONG lSemInitial,  // початкове значення семафора
  LONG lSemMax,  // максимальне значення семафора
  LPCTSTR lpSemName)  // вказівник на ім'я семафора
```

Для збільшення значення лічильника семафора використовується функція

```
BOOL ReleaseSemaphore (
  HANDLE hSemaphore,  // дескриптор семафора
  LONG cReleaseCount,  // величина збільшення лічильника
  // семафора
  LPLONG lpPreviousCount)  // вказівник на попереднє
  // значення
```

Коли значення лічильника семафора є більшим від нуля, семафор дає дозволений сигнал. Для очікування дозволеного сигналу семафорів використовуються функції WaitForSingleObject(...) і WaitForMultipleObjects(...). Вони зменшують значення лічильника семафора на одиницю при проходженні потоку.

Для створення об'єкта події призначена функція

```
HANDLE CreateEvent (
  LPSECURITY_ATTRIBUTES lpsa,
  // вказівник на структуру атрибутів захисту
  BOOL bManualReset,  // true, якщо подія скидається вручну
  BOOL bInitialState,  // true, якщо початковий стан події є
  // сигнальним
  LPTCSTR lpEventName)  // вказівник на ім'я об'єкта події
```

Потік може встановити подію в сигнальний стан, використовуючи функцію

```
BOOL SetEvent (
  HANDLE hEvent)  // дескриптор об'єкта події
```

Для очікування подій призначені функції WaitForSingleObject(...) і WaitForMultipleObjects(...).

Один потік може припинити виконання іншого потоку викликом функції

```
BOOL TerminateThread (
  HANDLE hThread,  // дескриптор потоку
  DWORD dwExitCode)  // код завершення потоку
```

Для переведення потоку в стан очікування на заданий час (в мілісекундах) призначена функція:

```
VOID Sleep (DWORD dwMilliseconds)
```

Для створення потоку використовується функція CreateThread(...). Закриває дескриптор об'єкта функція CloseHandle(...).

Завдання

З допомогою функцій Windows API написати програму surname_lab8 (surname — прізвище студента), яка виконає моделювання наступної задачі з використанням конкретних параметрів з таблиці відповідно до номеру варіанту.

У пансіоні відпочивають і розмірковують n філософів, пронумеровані від 1 до n. У їдальні розташований круглий стіл, навколо якого розміщені n стільців, також пронумеровані від 1 до n. На столі знаходиться одна велика тарілка зі спагеті, яка поповнюється нескінченно, також там розставлені n тарілок, в які накладається спагеті, і n виделок, призначення яких є очевидним.

Для того щоб пообідати, філософ заходить до їдальні і сідає на стілець зі своїм номером. При цьому філософ зможе їсти тільки в тому випадку, якщо вільні дві виделки — справа і зліва від його тарілки. При виконанні цієї умови філософ піднімає одночасно обидві виделки і може поглинати їжу протягом часу $t1$. В іншому випадку, філософу доводиться чекати звільнення обох виделок.

Пообідавши, філософ кладе обидві виделки на стіл одночасно (одну — зліва, а другу — справа від себе) і йде. Наступного разу він приходить обідати через проміжок часу $t2$. Описаний процес триває 10 секунд. Скільки разів кожному філософу вдасться пообідати протягом 10 секунд?

Варіант	Задача
1	$n = 5$, $t1 = 1$ мс, $t2 = 10$ мс, для синхронізації застосувати критичну секцію.
2	$n = 5$, $t1 = 1$ мс, $t2 = 10$ мс, для синхронізації застосувати м'ютекс.
3	$n = 5$, $t1 = 1$ мс, $t2 = 10$ мс, для синхронізації застосувати семафори.
4	$n = 5$, $t1 = 1$ мс, $t2 = 10$ мс, для синхронізації застосувати об'єкти події.
5	$n = 6$, $t1 = 2$ мс, $t2 = 12$ мс, для синхронізації застосувати критичну секцію.
6	$n = 6$, $t1 = 2$ мс, $t2 = 12$ мс, для синхронізації застосувати м'ютекс.
7	$n = 6$, $t1 = 2$ мс, $t2 = 12$ мс, для синхронізації застосувати семафори.
8	$n = 6$, $t1 = 2$ мс, $t2 = 12$ мс, для синхронізації застосувати об'єкти події.
9	$n = 4$, $t1 = 3$ мс, $t2 = 15$ мс, для синхронізації застосувати критичну секцію.
10	$n = 4$, $t1 = 3$ мс, $t2 = 15$ мс, для синхронізації застосувати м'ютекс.
11	$n = 4$, $t1 = 3$ мс, $t2 = 15$ мс, для синхронізації застосувати семафори.
12	$n = 4$, $t1 = 3$ мс, $t2 = 15$ мс, для синхронізації застосувати об'єкти події.
13	$n = 7$, $t1 = 4$ мс, $t2 = 20$ мс, для синхронізації застосувати критичну секцію.
14	$n = 7$, $t1 = 4$ мс, $t2 = 20$ мс, для синхронізації застосувати м'ютекс.
15	$n = 7$, $t1 = 4$ мс, $t2 = 20$ мс, для синхронізації застосувати семафори.
16	$n = 7$, $t1 = 4$ мс, $t2 = 20$ мс, для синхронізації застосувати об'єкти події.
17	$n = 8$, $t1 = 5$ мс, $t2 = 25$ мс, для синхронізації застосувати критичну секцію.

Варіант	Задача
18	$n = 8$, $t1 = 5$ мс, $t2 = 25$ мс, для синхронізації застосувати м'ютекс.
19	$n = 8$, $t1 = 5$ мс, $t2 = 25$ мс, для синхронізації застосувати семафори.
20	$n = 8$, $t1 = 5$ мс, $t2 = 25$ мс, для синхронізації застосувати об'єкти події.
21	$n = 9$, $t1 = 6$ мс, $t2 = 30$ мс, для синхронізації застосувати критичну секцію.
22	$n = 9$, $t1 = 6$ мс, $t2 = 30$ мс, для синхронізації застосувати м'ютекс.
23	$n = 9$, $t1 = 6$ мс, $t2 = 30$ мс, для синхронізації застосувати семафори.
24	$n = 9$, $t1 = 6$ мс, $t2 = 30$ мс, для синхронізації застосувати об'єкти події.
25	$n = 10$, $t1 = 2$ мс, $t2 = 13$ мс, для синхронізації застосувати критичну секцію.

Список рекомендованої літератури

1. Глинський Я. М., Анохін В. Є., Ряжська В. А. C++ і C++ Builder. Львів : СПД Глинський, 2011. 190 с.
2. Ковалев И. В., Кузнецов А. С., Царев Р. Ю. Операционные системы. Системное программное обеспечение. Красноярск, 2008. 120 с.
3. Коноваленко І. В., Федорів П. С. Системне програмування у Windows з прикладами на Delphi. Тернопіль : ТНТУ ім. І. Пулюя, 2012. 320 с.
4. Харт Д. М. Системное программирование в среде Windows ; пер. с англ. Москва : Вильямс, 2005. 592 с.
5. Programming reference for the Win32 API. URL : https://docs.microsoft.com/en-us/windows/win32/api.

Підписано до друку 15.06.2022. Формат 60×84/16.
Папір офсетний. Друк цифровий.
Гарнітура Computer Modern Roman. Умовн. друк. арк. 1,6.
Тираж 100. Зам. № 72 від 15.06.2022.

Віддруковано: Приватний підприємець Голіней О. М.
76018, м. Івано-Франківськ,
вул. Галицька, 128,
тел.: (0342) 58-04-32.