

ЖДТУ	Міністерство освіти і науки України Житомирський державний технологічний університет
------	---

Затверджено науково-методичною  
радою ЖДТУ  
протокол від «\_\_» \_\_\_\_\_ 20\_\_ р. №\_\_

**МЕТОДИЧНІ РЕКОМЕНДАЦІЇ**  
для виконання лабораторних робіт  
з навчальної дисципліни  
**«СИСТЕМНЕ ПРОГРАМУВАННЯ»**

для студентів освітнього рівня «бакалавр»  
денної та заочної форми навчання  
спеціальності 121 «Інженерія програмного забезпечення»  
освітньо-професійна програма «Інженерія програмного забезпечення»  
Факультет інформаційно-комп'ютерних технологій  
кафедра інженерії програмного забезпечення

Розглянуто і рекомендовано  
на засіданні кафедри інженерії  
програмного забезпечення  
протокол від «\_\_» \_\_\_\_\_ 201\_\_ р.  
№ \_\_\_\_

Розробник: ст. викл., кафедри інженерії програмного забезпечення Власенко О.В.

Житомир  
2018 – 2019 н.р.

## ЗМІСТ

Передмова

Вимоги до звіту

Вимоги до оформлення програм

Лабораторна робота №1 "Програми копіювання файлів"

Лабораторна робота №2 "Кодування файлу"

Лабораторна робота №3 "HelloWin"

Лабораторна робота №4 "KeyLock"

Лабораторна робота №5 "Миша"

Лабораторна робота №6 "Таймер"

Лабораторна робота №7 "Процеси та потоки"

Лабораторна робота №8 "Виняткові ситуації"

## ПЕРЕДМОВА

Дане видання не є учбовим посібником по системному програмуванню. Він включає перелік вимог, яким повинні задовольняти знання студентів перед виконанням ними певної лабораторної роботи, та деякі моменти, на які необхідно звернути увагу в ході роботи.

Звіт про виконання лабораторної роботи повинен бути зданий не пізніше ніж на наступному занятті.

---

## Вимоги до звіту

В різних ЛР вимоги до звіту можуть варіюватися, але любий звіт повинен складатися з частин, які відповідають нижче наведеним вимогам:

- описати програму та її призначення;
- вказувати типи та призначення глобальних змінних;
- описати методи та прийоми використані при вирішенні поставленої задачі;
- описати структуру програми.

В лабораторних роботах, які потребують розбору функцій, розбір виконувати за наступним шаблоном:

- *тип поверненого значення;*
- *що повернене значення означає;*
- *які параметри отримує функція, їх типи, що означають;*
- *що виконує функція;*
- *особливості;*

Якщо функція в якості параметру отримує одну з констант, описану в header файлі, необхідно перерахувати всі споріднені з нею константи і описати їх.

## Приклад:

### Синтаксис :

```
#include <fcntl.h>
#include <sys/stat.h>
int open(const char *filename, int access [, unsigned mode]);
```

Файл, який включає прототип - *io.h*

### Описання:

Функція *open* відкриває файл, ім'я котрого задано параметром *filename*, і підготовлює його до наступних операцій читання та / або запису, в залежності від параметру *access*.

Для утворення файлу в звичайному режимі, ви можете присвоїти відповідні значення *ifmode*, або при відкритті файла вказати опції *O\_CREAT* і *O\_TRUNC* зв'язані операцією побітового логічного додавання (*OR*) з необхідним режимом передачі.

### Наприклад:

```
open("xmp",O_CREAT|O_TRUNC|O_BINARY,S_IREAD);
```

створює двійковий файл з атрибутом тільки для читання з ім'ям *XMP*, причому, Якщо він

<b>ЖДТУ</b>	<b>Міністерство освіти і науки України</b> <b>Житомирський державний технологічний університет</b>
-------------	---

вже існує, то він обрізається до нульової довжини.

Для функції `open` параметр `access` конструюється шляхом логічного побітового додавання прапорів, перерахованих в двох наступних списках. З першого списку може бути використаний тільки один прапор; інші прапори можуть використовуватися в любых логічних комбінаціях.

**Список 1:** прапори читання / запису .

`O_RDONLY` відкриття тільки для читання .

`O_WRONLY` відкриття тільки для запису .

`O_RDWR` відкриття для читання і запису .

**Список 2:** інші прапори доступу.

`O_NDELAY` Не використовується; для сумісності з системою UNIX.

`O_APPEND` Якщо прапорець встановлений, то перед кожною операцією запису, вказівник файлу буде встановлюватися на кінець файлу.

`O_CREAT` Якщо файл існує, цей прапорець не має ніякого значення. Якщо файл не існує, він буде створений, і біти з аргументу `mode` будуть використані для установки бітів - атрибутів файлу, як і в функції `chmod`.

`O_TRUNC` Якщо файл існує, його довжина усікається до 0. Атрибути файлу остаються не змінними.

`O_EXCL` використовується тільки разом з `O_CREAT`. Якщо файл уже існує, то функція повертає помилку.

`O_BINARY` Даний прапор може бути встановлений для гарантованого відкриття файлу в двійковому режимі.

`O_TEXT` Даний прапор може бути встановлений для гарантованого відкриття файлу в текстовому режимі.

Ці константи (`O_...`) визначені в файлі `fcntl.h`.

Якщо ні `O_BINARY`, ні `O_TEXT` не вказані, файл відкривається в режимі трансляції, відповідно глобальній змінній `ifmode`.

Якщо в побудові параметру `access` бере участь прапор `O_CREAT`, вам необхідно вказати аргумент `mode` з наступних символічних констант, визначених в файлі `sys/stat.h`.

Значення параметру доступу `mode`:

`S_IWRITE` Дозвіл на запис.

`S_IREAD` Дозвіл на читання.

`S_IREAD/S_IWRITE` Дозвіл на читання / запис.

Повернені значення: При успішному завершенні `open` повертає ціле невід'ємне число `handle` - логічний номер відкритого файлу. Вказівник файлу (Вказівник поточної позиції) встановлюється на начало файлу. При помилці функція повертає значення `-1`, і змінна `errno` отримує одне з наступних значень:

`ENOENT` - Маршрут або ім'я файлу не знайдені;

`EMFILE` - Занадто багато відкритих файлів;

`EACCESS` - Доступ заборонено;

`EINVA` - Невірний код доступу.

## Вимоги до оформлення програм

Тексти програм, написаних студентом, повинні відповідати наступним вимогам:

1. Кожний файл з текстом програми повинен мати заголовок який включає:

- ім'я файлу;

- коротку ремарку яка пояснює призначення файлу;
- рядок який ідентифікує автора.

2. В кінці файлу має бути рядок, який вказує, що це кінець файлу.

3. Вирівнювання тексту повинно відповідати наступним умовам:

- Директиви препроцесора, об'явлення прототипів функцій, глобальних змінних - вирівнюються в притулок до лівого краю.
- Кожний оператор або блок операторів, логічно вкладений в інший, зсовується на дві позиції вправо відносно зовнішнього. Фігурна дужка ({}), яка означає початок блоку, повинна знаходитись в тому ж рядку, що і оператор, який вміщує цей блок, а дужка, яка означає кінець блоку (}) - в тому ж стовпчику, що починається цей оператор

**приклад:**

```
if(a){
  while(i<0){
    i++;
    j++;
    a = 0;
  }
}
```

4. Функції та змінні повинні мати осмислені імена які відповідають угорській нотації:

Префікс	Тип даних
A	Масив (Складний тип)
Ch	Символ
Cb	Лічильник байтів
Dw	Довге ціле без знаку
H	Логічний номер
Hdc	Логічний номер контексту пристрою
Hwnd	Логічний номер вікна
I	індекс (Складний тип)
L	Довге ціле
Lp	Дальній (довгий) покажчик (Складний тип)
N	Ціле
Np	Ближній (короткий) покажчик
Pt	Точка описана парою координат x,y
Sz	Рядок який закінчується нулем
W	Ціле без знаку

<b>ЖДТУ</b>	<b>Міністерство освіти і науки України Житомирський державний технологічний університет</b>
-------------	---

1. *Касаткін А.І.* Управление ресурсами. - Минск: Высшая школа, 1992.
2. *Касаткін А.І.* Системное программирование. - Минск: Высшая школа, 1991.
3. Власенко О.В., Данильченко О.М., Северин О.О. Системне програмування. Курс лекцій. Частина 1. (бібліотека ЖІТІ)

### **Лабораторна робота №1 "Програми копіювання файлів"**

#### **Література до перших трьох ЛР**

1. *Касаткін А.І.* Управление ресурсами. - Минск: Высшая школа, 1992.
2. *Касаткін А.І.* Системное программирование. - Минск: Высшая школа, 1991.
3. Власенко О.В., Данильченко О.М., Северин О.О. Системне програмування. Курс лекцій. Частина 1. (бібліотека ЖІТІ)

#### **Підготовка:**

1. Розібрати всі функції використані в програмі.
2. Опрацювати теоретичний матеріал.

#### **Завдання:**

1. Навчитися користуватись програмами `copy1.exe` та `copy2.exe` ( архів [lab1.rar](#)).
2. Розібрати роботу програм.
3. В програмі `copy1.c` замінити бінарний режим доступу до обох файлів на текстовий і запустити програму для копіювання досить великого бінарного файлу. Результати експерименту зафіксувати в зошиті. Пояснити причину ефекту.
4. В програмі `copy1.c` замінити бінарний режим доступу до файлу, який записується, на текстовий і запустити програму. Результати експерименту зафіксувати в зошиті. Пояснити причину ефекту.
5. В програмі `copy2.c` знайти оптимальний розмір буферу. Обґрунтувати вибір.
6. В програмі `copy2.c` модифікувати програму так, щоб вона виводила на екран вміст файлу за допомогою функцій `puts()`, `fputs()`, `printf()`, `fwrite()`. Результати експерименту зафіксувати в зошиті. Пояснити причину ефекту.

#### **Контрольні запитання:**

1. Чим відрізняється текстовий режим доступу до файлу від бінарного?
2. Що таке дескриптор(`handle`) файлу?
3. Чим відрізняється префіксний доступ до файлу від потокового?
4. В яких випадках більше доцільний потоковий доступ? Чому?

### **Лабораторна робота №2 "Кодування файлу"**

#### **Література до перших трьох ЛР**

1. *Касаткін А.І.* Управление ресурсами. - Минск: Высшая школа, 1992.
2. *Касаткін А.І.* Системное программирование. - Минск: Высшая школа, 1991.

ЖДТУ	Міністерство освіти і науки України Житомирський державний технологічний університет
------	---

3. Власенко О.В., Данильченко О.М., Северин О.О. Системне програмування. Курс лекцій. Частина 1. (бібліотека ЖІТІ)

**Підготовка:**

1. Повторити ЛР №1.
2. Розібрати функції *lseek()*, *fseek()*
3. Опрацювати теоретичний матеріал.

**Технічне завдання:**

Програма повинна забезпечувати шифруванні і дешифруванні будь яких файлів по довільному алгоритму з використанням пароля.

**Вимоги до програми:**

1. Назва програми: **Encode**.
2. **Інтерфейс** - командний рядок, в який вводяться режим роботи програми, пароль, а також імена файлів.

**encode** *</e |/d>* *<File\_to\_code>* *[distination\_file]* *[/p=password]*

Наприклад:

**encode /e myfile.txt myfile.cod /p=mypassword**, де

*/e* або */d* - ключ який визначає шифрування або дешифрування;

**myfile.txt** - специфікація файлу який має бути зашифрований;

**myfile.cod** - специфікація файлу в який має бути записано зашифровану послідовність;

**/p=mypassword** - пароль для шифрування.

3. Режим *шифрування* - ключ */e*, режим *дешифрування* - ключ */d*.

4. При невірному завданні параметрів програма повинна виводити інформацію про вірне завдання параметрів.

Наприклад: **encode** *</e |/d>* *<File\_to\_code>* *<distination\_file>* *[/p=password]*

5. Зашифрований файл повинен складатися із *заголовку* і *кодованих даних*.

6. Заголовок повинен містити:

*сигнатуру виду файлу;*

*номер версії програми;*

*рядок Copyright*, із якого зрозуміло, якою програмою файл був зашифрований;

*контрольну суму*, що повинна використатися в алгоритмі дешифрування;

*імя вихідного файлу* для його відновлення при дешифруванні

*іншу інформацію* на розсуд розробника.

Для роботи з заголовком краще використати структурну змінну. Наприклад:

```
struct Header {
    char signat[3];
    int version;
    char CopyRight[30];
    unsigned long CRC;
    char filename[13];
    ...
}
```

7. Необхідно рахувати контрольну суму, яка буде контролювати правильність дешифрування.

Найпростіший метод підрахунку контрольної суми - це сумування всіх байтів файлу.

Наприклад:

```
unsigned long CRC = 0;
...
while(!eof(source_file)) {
...
    CRC +=buffre[i];
...
}
```

Для виконання кодування даних можна використати будь який алгоритм. Найпростішим може бути алгоритм Гамування (гаммирования).Базується він на бітовій операції XOR. В мосі Сі операція визначена символом '^'. Ця операція є зворотньою. Тоб то:

```
a xor b = c
c xor b = a
c xor a = b.
```

Використовуючи це правило в программі можна записати це так:

```
int j = 0;
for( int i = 0; i < count_byte_in_buffer; i++ ) {
    if( j == strlen( password ) ) j = 0;
    buffer[ i ] ^= password[ j++ ];
}
```

Результат буде будуватись по наступній схемі:

Вхідна послідовність	b1	b2	b3	b4	b5	b6	b7	b8	b9	b10	...	bn
Символи паролю "vasya"	v	a	s	y	a	v	a	s	y	a	...	...
Вихідна послідовність	c1=b1^'v'	c2=b2^'a'	c3=b3^'s'	c4	c5	c6	c7	c8	c9	c10	...	cn

Процес дешифрування проходить по тій же схемі.

**Контрольні запитання:**

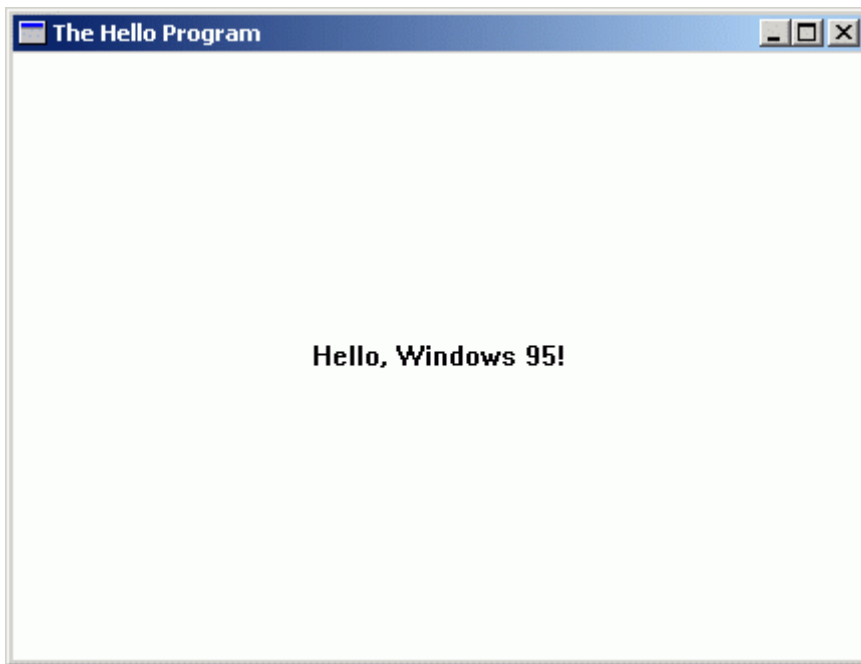
1. Які існують режими доступу до файлу?
2. В якому режимі необхідно відкрити файл, щоб мати можливість записати контрольну суму на початку файлу з даними?
3. Що буде в вихідному файлі, якщо вхідний містить 100 нульових байтів? Як уникнути цього?
4. Які ще дії над байтами вхідної послідовності можна булоб зробити без загублення даних?

**Лабораторна робота №3 "HelloWin"**

**Завдання для виконання.**

1. Розібрати текст програми helloworldin.c та вивчити матеріал викладений у файлі допомоги (архів [lab3.rar](#))





2. Створити аналогічну програму, яка друкує в вікно:

- Змінні середовища програми (Environment)
- Шлях - звідки була запущена на виконання програма

### Контрольні питання.

1. Чим відрізняється програма для Windows від програми для DOS?
2. Яка функція є точкою входу в програму?
3. Що таке клас вікна?
4. Що таке віконна функція? Віконна процедура?
5. Навіщо виконувати реєстрацію вікна?
6. Як зв'язана віконна функція з вікном?
7. Що таке цикл обробки повідомлень? Які його функції?
8. Що таке повідомлення? Хто надсилає повідомлення?
9. Кого отримує повідомлення для вікна?
10. Які повідомлення ви вже знаєте? Що визначає кожне з них?
11. Що таке контекст пристрою? Яким чином виконується виведення в вікно програми?

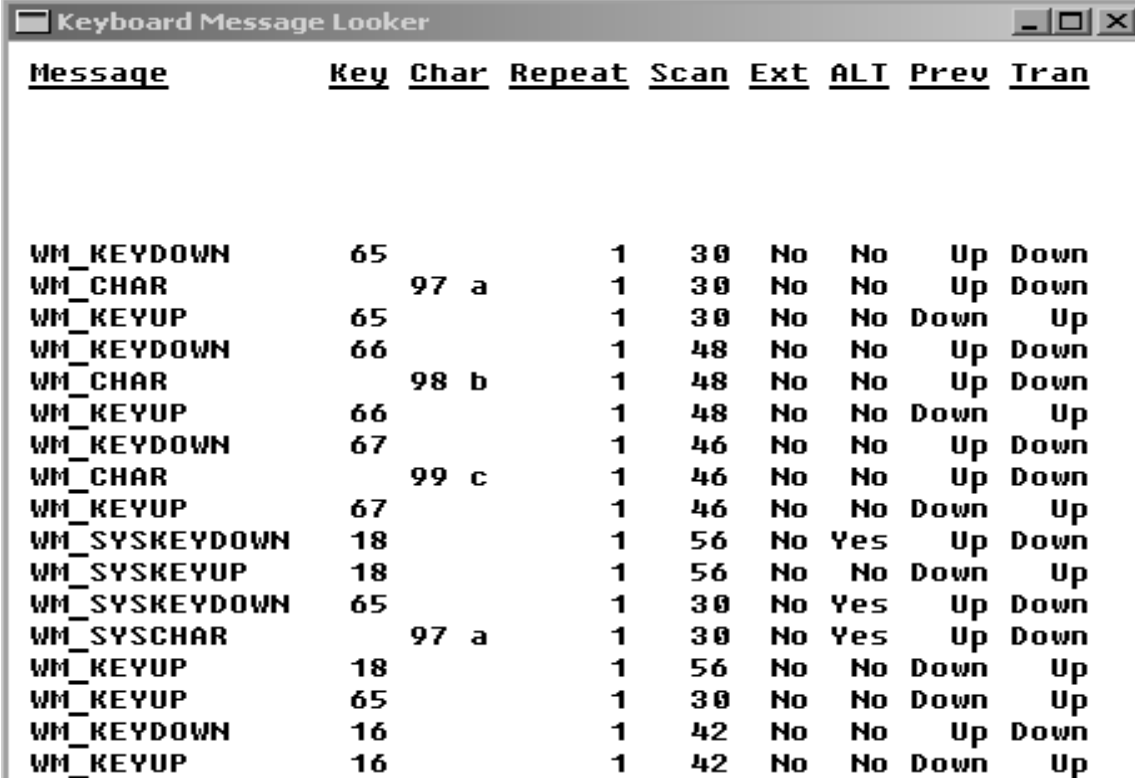
### Лабораторна робота №4 "KeyLock"

1. Вивчити матеріал викладений у файлі допомоги (Опис роботи з клавіатурою, архів [lab4.rar](#))
2. Створити програму, яка друкує в вікно інформацію про натискання клавіш на клавіатурі в подібну таблицю:

Massege	Key	Char	Repeat	Scan	Ext	Alt	Prev	Tran
---------	-----	------	--------	------	-----	-----	------	------

За основу (каскад програми) можна використати програму з лабораторної роботи №4. До неї потрібно додати обробку хочаб одного повідомлення WM\_CHAR.

Приклад роботи програми показано нижче на малюнку. Приклад готової програми в архіві матеріалу до лабораторної роботи (Програма keylock.exe).



<u>Message</u>	<u>Key</u>	<u>Char</u>	<u>Repeat</u>	<u>Scan</u>	<u>Ext</u>	<u>ALT</u>	<u>Prev</u>	<u>Tran</u>
WM_KEYDOWN	65		1	30	No	No	Up	Down
WM_CHAR		97 a	1	30	No	No	Up	Down
WM_KEYUP	65		1	30	No	No	Down	Up
WM_KEYDOWN	66		1	48	No	No	Up	Down
WM_CHAR		98 b	1	48	No	No	Up	Down
WM_KEYUP	66		1	48	No	No	Down	Up
WM_KEYDOWN	67		1	46	No	No	Up	Down
WM_CHAR		99 c	1	46	No	No	Up	Down
WM_KEYUP	67		1	46	No	No	Down	Up
WM_SYSKEYDOWN	18		1	56	No	Yes	Up	Down
WM_SYSKEYUP	18		1	56	No	No	Down	Up
WM_SYSKEYDOWN	65		1	30	No	Yes	Up	Down
WM_SYSCHAR		97 a	1	30	No	Yes	Up	Down
WM_KEYUP	18		1	56	No	No	Down	Up
WM_KEYUP	65		1	30	No	No	Down	Up
WM_KEYDOWN	16		1	42	No	No	Up	Down
WM_KEYUP	16		1	42	No	No	Down	Up

### Лабораторна робота №5 "Миша"

Програма обробки повідомлень від миші.  
Підрахунок відстані та швидкості руху миші.

Завдання:

1. Вивести координати миші у вікні.
2. Знайти відстань між двома точками у вікні.
3. Знайти швидкість руху миші на проміжку.

Для виконання цієї лабораторної роботи, необхідно вивчити теоретичний матеріал.

1. Виведення координат миші у вікні.

Для того щоб визначити координати миші, необхідно обробити повідомлення WM\_MOUSEMOVE, яке приходить кожний раз, коли виконується рух миші при активному вікні, в камках вікна. Параметрами цього повідомлення є координати миші.

Обробляючи це повідомлення, ви формуєте текстовий рядок наприклад наступним чином:

```
X = LOWORD(lParam);  
Y = HIWORD(lParam);  
wsprintf(str, " X = %d   Y = %d ", x,y );
```

## 2. Відстань між двома точками у вікні.

Необхідно зафіксувати дві точки. Нехай це будуть точки в яких користувач натисне ліву а потім праву кнопку миші.

Коли виконується натискання лівої або правої кнопки миші, ми отримуємо повідомлення:

*WM\_LBUTTONDOWN* або *WM\_RBUTTONDOWN*

Разом з цими повідомленнями передаються координати миші.

```
X = LOWORD(lParam);  
Y = HIWORD(lParam);
```

Це координати в яких було виконане натискання.

Нехай ліва кнопка миші була натиснута в координатах (X1,Y1) а права в координатах (X2,Y2). Відстань між двома точками можна отримати за виразом:

$$S = \text{sqrt}((X2 - X1) * (X2 - X1) + (Y2 - Y1) * (Y2 - Y1));$$

Для визначення швидкості руху миші нам потрібно знати час руху на проміжку.

Для визначення часу, потрібно зафіксувати час при натисканні лівої кнопки миші, а потім час при натисканні правої кнопки миші.

```
T1 = clock(); // цей рядок пишемо в обробник повідомлення WM_LBUTTONDOWN
```

...

```
T2 = clock(); // цей рядок пишемо в обробник WM_RBUTTONDOWN
```

```
DT = T2 - T1;  
Speed = S / DT
```

Тепер залишається тільки вивести значення в вікно.

Для виведення отриманих текстових рядків можна використати функцію TextOut(). Вона вигідна тим, що дозволяє виводити в визначені координати вікна.

Виведення отриманих рядків необхідно робити тільки в обробці повідомлення WM\_PAINT. Нагадую, воно приходить тоді, коти операційна система вважає, що інформація в нашому вікні застаріла.

Якщо, нам потрібно згенерувати повідомлення WM\_PAINT, то викликаємо функцію InvalidateRect(), яка повідомляє ОС про те що якась частина нашого вікна (або все

<b>ЖДТУ</b>	<b>Міністерство освіти і науки України</b> <b>Житомирський державний технологічний університет</b>
-------------	---

вікно) застаріло. Виклик цієї функції ми робимо кожний раз, коли необхідно перемалювати вікно.

### **Додаткове (ускладнене завдання).**

Знайти **миттєву швидкість** руху миші.

Підказка. Використовуєте програмований таймер, для отримання фіксованих проміжків часу, на яких визначаєте відстань пройдену мишею, вона й буде швидкістю руху.

### **Лабораторна робота №6 "Таймер"**

#### **Програмування таймеру.**

**Робота з системною датою та часом.**

**Створення декількох таймерів, та обробка повідомлень від таймеру.**

У даній лабораторній роботі вам необхідно вивчити роботу « таймера ».

Таймер - це об'єкт ядра ОС Windows , який може бути ініціалізованим першим , після чого в чергу повідомлень для вашої програми буде приходити повідомлення WM\_TIMER щоразу після закінчення заданого проміжку часу.

Для установки таймера використовується функція WinApi : SetTimer ( ) .

Для знищення таймера використовують KillTimer ();

Для написання 7 лаб. роботи необхідно взяти за основу , 4 лаб. роботу . У ній створюється вікно , в яке необхідно вивести наступне:

1 . Системну дату .

Для цього вам необхідно її отримати . Це можна зробити , викликавши функцію:

GetDateFormat ();

Виклик цієї функції краще виконати у обробці повідомлення WM\_CREATE , тобто при створенні вікна . Результатом роботи функції буде заповнена текстова рядок з поточною датою , яку ви і виведете при обробці повідомлення WM\_PAINT .

2 . Системний час з точністю до секунд .

Так як час буде змінюватися кожену секунду , то його і оновлювати необхідно буде кожену секунду.

Тому необхідно використовувати таймер. Наприклад , його можна створити таким викликом:

SetTimer ( hwnd , 6 , 1000, NULL ) ;

де

hwnd - дескриптор вікна ;

6 (другий параметр) - ідентифікатор таймера , чому я поставив йому номер 6 а не 1 зрозумієте нижче. Але це просто його номер.

1000 ( третій параметр) - проміжок часу, задається в мсек . 1000 мсек = 1 сек.

NULL (четвертий параметр) - покажчик на функцію , яка буде викликана , для обробки повідомлення WM\_TIMER , якщо задано NULL , то викликається ваша віконна функція .

Виклик цієї функції краще виконати при обробці повідомлення WM\_CREATE .  
Тут же, не забудьте вставити зворотний виклик , тобто знищення таймера.

KillTimer ( hwnd , 6 ) ;

Де другий параметр це ідентифікатор видаляється таймера.

Виклик цієї функції краще зробити в обробці повідомлення WM\_DESTROY , тобто при знищенні вікна .

Встановивши таймер , ви будете отримувати повідомлення WM\_TIMER , кожна секунду.

У обробнику цього повідомлення вставте виклик функції

GetTimeFormat ( ) , яка вам поверне системний час . Результат буде в текстовій рядку , висновок якої вставте в обробці WM\_PAINT .

3 . Програмування п'яти таймерів на різний час.

Виведіть у вікно п'ять рядків наступного змісту:

Timer number 1 ( 5s. ) - OFF

Timer number 2 ( 10s. ) - OFF

Timer number 3 ( 15s. ) - OFF

Timer number 4 ( 20s. ) - OFF

Timer number 5 ( 25s. ) - OFF

Якщо користувач виконати клацання лівої кнопки миші по якій те з п'яти рядків , то фраза OFF , повинна змінитися на фразу ON , і після чого , через відповідний проміжок часу ( 5 , 10 , 15 , 20 або 25 сек. ) Відповідна рядок буде виглядати наприклад так:

Timer number 4 ( 20s. ) - ON ( BEEP! )

Фраза BEEP , повинна з'являтися кожен раз після закінчення проміжку часу сигналізуючи про те що даний таймер включений і працює. Не забудьте її прибирати для того щоб було видно момент її включення наступного разу.

Якщо користувач виконає повторне клацання лівою кнопкою миші по тому ж таймером , то він повинен вимкнутися ( ON -> OFF ) , і після чого сигнал BEEP , не повинен біля нього з'являтися .

Користувач може включити будь-яку кількість таймерів (він 1 до 5 одночасно ) , і всі вони повинні «робити» BEEP , після закінчення свого проміжку часу.

Для виконання останнього ( третього) завдання , зробіть наступне:

Вставте обробку повідомлення WM\_LBUTTONDOWN , яке приходить , при натисненні лівої кнопки і миші. Перевірте координати Миші , в яких виконано натискання , є координати збігаються з одного з рядків 5 таймерів , то виконайте дії з встановлення даного таймера.

Наприклад для першого SetTimer ( hwnd , 1 , 5000 , NULL ) ;

Або для другого - SetTimer ( hwnd , 2 , 10000 , NULL ) ;

Перемалуйте OFF на ON .

Якщо таймер був включений , то вимкніть його

ЖДТУ	Міністерство освіти і науки України Житомирський державний технологічний університет
------	---

KillTimer ( hwnd , 1);  
Перемалюйте ON на OFF.

Як тільки ви встановите черговий таймер , то він теж стане надсилати WM\_TIMER . Для того щоб розрізнити від якого таймера прийшло повідомлення , тоєсть який проміжок часу пройшов , для цього скористайтеся номером таймера який ви отримуєте разом з повідомленням WM\_TIMER в параметрі lParam .

Таким чином в обробнику WM\_TIMER : можна виконати наступне:

```
switch ( lParam ) {
case 1: // обробка таймера на 5 сек
break ;
case 2: // обробка таймера на 10 сек
break ;
case 3: // обробка таймера на 15 сек
break ;
case 4: // обробка таймера на 20 сек
break ;
case 6 : // обробка таймера на 1 сек формування рядка системного часу
break ;
}
```

Дана лаб.раб . розрахована на одне лабораторне заняття .

## Лабораторна робота №7 "Процеси та потоки"

**Взаємодія між процесами.**

**Розподіл даних між процесами.**

**Робота з файлами які відображаються у пам'ять.**

Необхідно написати дві програми , які матимуть спільні дані .

Існує кілька механізмів реалізації спільного доступу до даних різних процесів .

Скористаємося одним з них , найбільш зручним - проєціюванням файлу в пам'ять.

Одна програма буде сортувати дані у файлі , а інша відображати вміст цього файлу.

Працювати обидва процеси будуть одночасно .

Створіть файл data.dat . У ньому записані числа , згенеровані випадковим чином. Кількість чисел - 20-30 штук. Діапазон значень : від 10 до 100. ( Це саме числа , а не символічні рядки храненияіASCII коди цифр !)

Програма № 1 . « Сортування даних»

1 . Беремо за основу програму лаб.раб № 3 .

2 . Включаємо обробку події натискання клавіші, і відстежуємо в ньому натискання пробілу. Якщо користувач натиснув пробіл , значить починаємо сортування даних.

3 . Виконуємо проектування файлу в пам'ять. Використовуємо для цього створений файл data.dat . В результаті отримуємо доступ до даних як до звичайного одновимірного масиву .

ЖДТУ	Міністерство освіти і науки України Житомирський державний технологічний університет
------	---

4 . Виконуємо сортування масиву , будь-яким з методів сортування. Вставте 1-но секундну затримку для кожної ітерації сортування масиву , це дозволить потім наглядней побачити процес сортування.

5 . По закінченню сортування , програма виводить у вікно , рядок « Робота завершена».

Програма № 2. « Висновок файлу даних у вікно »

1 . Беремо за основу програму лаб.раб № 3 .

2 . Виконуємо проектування файлу в пам'ять. Використовуємо для цього створений файл data.dat . В результаті отримаємо доступ до даних як до звичайного одновимірного масиву . Цей же файл проектує на згадку попередня програма.

3 . Створюємо таймер на 0.5 секунди . При отриманні повідомлення від таймера , виконуємо висновок всього масиву у вікно. Передбачте коректний перевивод даних у вікно , без накладень . У вікно виводиться не числа з масиву , а рядки одного і того ж символу , наприклад « \* » , в кількості рівній числу з масиву.

Запускаємо на виконання обидві програми одночасно. Коли друга програма запустилася і виконує виведення даних у вікно (виводить поки одну і ту ж саму картинку кожні пів секунди ) , натискаємо пробіл у першій програмі і вона починає сортувати масив . При цьому , так як вони дані беруть з одного і того ж файлу (обидві проектували його собі на згадку ) , то перша вносить зміни переставляючи дані при сортуванні , а друга виводить з себе у вікно і ми бачимо хід процесу сортування . Тимчасову затримку у першій програмі можна при потребі збільшити.

Ці дві програми демонструють можливість організації спільного доступу процесів до одних і тих же даних . Так само демонструється механізм проектування файлу в пам'ять , як один з найкращих методів доступу до файлу.

Завдання розраховане на 2 лабораторних заняття .

Теорія по проєцированию файлів у Методичці (частина 3).

### **Лабораторна робота №8 "Виняткові ситуації"**

**Критичні секції коду програми.**

**Виняткові ситуації.**

**Виключення при взаємодії процесів.**