МЕТОДИЧНІ ВКАЗІВКИ

до виконання курсової роботи

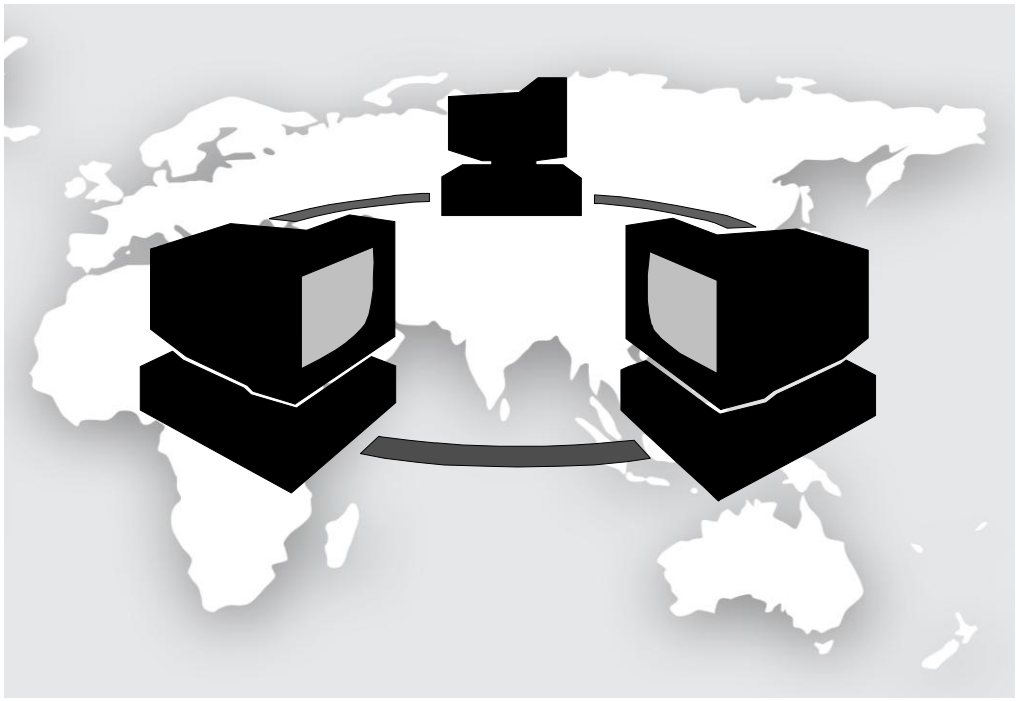
з дисципліни

"Прикладне програмування"

для студентів спеціальності

"Комп’ютерна інженерія"

всіх форм навчання



Міністерство освіти і науки України
Вінницький національний технічний університет

МЕТОДИЧНІ ВКАЗІВКИ
до виконання курсової роботи з дисципліни "Прикладне програмування" для студентів напряму підготовки "Комп'ютерна інженерія" всіх форм навчання

Вінниця
ВНТУ
2016

УДК 681.3.062(075)
ББК 32.972.5

Рекомендовано до друку Методичною радою Вінницького національного технічного університету Міністерства освіти і науки України (протокол № 3 від 20.11.2016 р.)

Рецензенти:
Ю. В. Булига, кантидат технічних наук, доцент кафедри ГМ
В. П. Майданюк, кандидат технічних наук, доцент кафедри ПЗ

Методичні вказівки до виконання курсової роботи з дисципліни "Прикладне програмування" для студентів спеціальності "Комп'ютерна інженерія" всіх форм навчання / Уклад. О. Д. Азаров, О. І. Черняк, Л. А. Савицька – Вінниця : ВНТУ, 2016 – 43 с.

У методичних вказівках викладено вимоги до змісту розділів пояснювальної записки до курсової роботи та її оформлення. Наведено завдання до курсової роботи, зразки оформлення титульного аркуша та індивідуального завдання. Надано список літератури. Методичні вказівки можуть використовуватись для виконання контрольних робіт студентами заочної форми навчання та для самостійної роботи студентів.

ЗМІСТ

ВСТУП 4
1 КОРОТКІ ТЕОРЕТИЧНІ ВІДОМОСТІ 5
1.1 Поштові скриньки 5
1.2 Іменовані канали 8
1.3 Сокети windows 13
2 ЗАВДАННЯ ДО КУРСОВОЇ РОБОТИ 25
2.1 Загальні вимоги 25
2.2 Варіанти завдань 25
3 РЕКОМЕНДАЦІЇ ЩОДО ОФОРМЛЕННЯ КУРСОВОЇ РОБОТИ 32
3.1 Правила оформлення пояснювальної записки 32
3.2 Структура пояснювальної записки 32
ПІСЛЯМОВА 35
ЛІТЕРАТУРА 37
ДОДАТОК А 38
ДОДАТОК Б 39
ДОДАТОК В 41

ВСТУП

Дані методичні вказівки призначені для виконання курсової роботи з дисципліни "Прикладне програмування".

У результаті виконання курсової роботи студенти повинні отримати практичні навички зі створення програм для роботи у мережі Windows на основі потокового та дайтаграмного передавання інформації з використанням поштових скриньок, іменованих каналів та сокетів.

Виконання даної курсової роботи сприяє засвоєнню студентами таких теоретичних питань: інформаційна структура протоколів обміну даними, аспекти клієнт-серверної організації програмного забезпечення, основи організації комп'ютерних мереж, аспекти обмеження доступу до ресурсів, аспекти програмування файлів і потоків, аспекти роботи поштових скриньок та іменованих каналів, принципи організації обміну даними за допомогою інтерфейсу прикладного програмування WINSOCK, аспекти використання адрес комп'ютерів та портів, види встановлення з'єднань між комп'ютерами та відповідні протоколи.

Для виконання курсової роботи необхідно знати принципи організації комп'ютерних мереж та техніки програмування мовою C++, а також мати практичні навички з програмування у середовищі Visual C++. Бажано мати доступ до комп'ютерної мережі і можливість виконувати програми хоча б на двох комп'ютерах у мережі. Проте, можна виконувати мережеві програми і на одному комп'ютері. Для цього потрібно у програмах вказувати локальну адресу. На комп'ютері повинна бути встановлена операційна система Windows і середовище програмування Visual Studio. Якщо на комп'ютері встановлені програми чи працюють служби, що обмежують можливість обміну інформацією у мережі, то потрібно зняти ці обмеження за допомогою встановлення відповідних параметрів або тимчасово відключити обмежувальні програмні засоби.

1 КОРОТКІ ТЕОРЕТИЧНІ ВІДОМОСТІ

У даному розділі наведено короткі теоретичні відомості про такі механізми обміну інформацією у мережах, як поштові скриньки, іменовані канали і сокети. Більш детальну інформацію можна отримати в [3,4].

1.1 Поштові скриньки

Поштові скриньки розташовуються в розділі оперативної пам’яті, іменованому як Mailslot. Основне обмеження поштових скриньок – вони допускають тільки ненадійне односпрямоване передавання повідомлень від клієнта до сервера. Основна перевага – клієнтські програми можуть легко посилати широкомовні повідомлення одному чи декільком серверним програмам.

Поштові скриньки використовують архітектуру клієнт-сервер, у якій дані передаються тільки від клієнта до сервера. Сервер створює поштову скриньку і може читати з неї дані. Клієнт відкриває існуючу поштову скриньку і може записувати у неї дані. Клієнтська і серверна програми використовують функції ReadFile і WriteFile для відправлення й одержання даних поштовою скринькою, а також правила іменування файлової системи Windows.

Поштові скриньки іменуються за таким правилом:

*сервер**Mailslot**[шлях]ім'я*

сервер – ім'я сервера, на якому створюється поштова скринька;

Mailslot – фіксоване обов'язкове ім’я;

\[*шлях*]*ім'я* – дозволяється задавати кілька рівнів каталогів.

Поле \\server може являти собою крапку (.), зірочку (*), ім'я домену чи сервера.

Для передавання повідомлень використовують дайтаграми без встановлення з'єднання, якщо розмір повідомлення не перевищує 424 байти. Повідомлення розміром більше 426 байтів передаються із встановленням з'єднання. Допускається з'єднання тільки одного клієнта з одним сервером. Навіть при встановленні з’єднання між комп’ютерами не гарантується, що повідомлення буде записано у поштову скриньку.

При написанні програми з використанням поштової скриньки потрібно під’єднати заголовний файл Winbase.h. або Windows.h.

Для реалізації сервера поштової скриньки потрібно виконати такі дії.

1. Створити описувач поштової скриньки за допомогою функції CreateMailslot.

2. Одержати дані від будь-якого клієнта шляхом виклику функції ReadFile з описувачем поштової скриньки як параметром.

3. Закрити описувач поштової скриньки за допомогою функції CloseHandle.

Функція ***CreateMailslot*** визначена так:

HANDLE CreateMailslot(

LPCTSTR lpName,
DWORD nMaxMessageSize,
DWORD lReadTimeout,
LPSECURITY_ATTRIBUTES lpSecurityAttributes);

lpName – задає ім'я поштової скриньки. Воно повинно мати такий вигляд: \\.\Mai1slot\[шлях]ім'я. Тут ім'я сервера замінене крапкою, що означає локальний комп'ютер, оскільки не можна створити поштову скриньку на віддаленому комп'ютері.

nMaxMessageSize – задає максимальний розмір (у байтах) повідомлення, що може бути записане у поштову скриньку.

lReadTimeout – задає кількість часу (у мілісекундах), протягом якого операції читання чекають вхідних повідомлень.

lpSecurityAttibutes – повинен мати значення NULL.

Лістинг 1.1 – Приклад сервера поштових скриньок

```
// Server.cpp
//
#include <windows.h>
#include <stdio.h>
void main(void)
{
    HANDLE Mailslot;
    char buffer[256];
    DWORD NumberOfBytesRead;
    // Створення поштової скриньки
    if ((Mailslot = CreateMailslot("\\\\.\\Mailslot\\Myslot", 0,
MAILSLOT_WAIT_FOREVER, NULL)) == INVALID_HANDLE_VALUE)
    {
        printf("Failed to create a mailslot %d\n", GetLastError());
        return;
    }
    while(ReadFile(Mailslot, buffer, 256, &NumberOfBytesRead, NULL)
    != 0)
    {
        buffer[NumberOfBytesRead]=0;
        printf("%s\n", buffer);
    }
    CloseHandle(Mailslot);
}
```

Клієнт поштової скриньки відкриває існуючу поштову скриньку і записує у неї дані. Для цього потрібно виконати такі дії.

1. Відкрити поштову скриньку за допомогою функції CreateFile.
2. Записати дані у поштову скриньку за допомогою функції WriteFile.

3. Закрити поштову скриньку за допомогою функції CloseHandle.

Клієнт відкриває поштову скриньку без встановлення зв'язку за допомогою функції *CreateFile* з такими параметрами:

lpFileName – ім'я поштової скриньки.

dwDesiredAccess – задати значення GENERIC_WRITE.

dwShareMode – задати значення FILE_SHARE_READ.

lpSecurityAttibutes – задати значення NULL.

dwCreationDisposition – задати значення OPEN_EXISTING.

dwFlagsAndAttributes – задати значення FILE_ATTRIBUTE_NORMAL,

hTemplateFile – задати значення NULL

Після успішного відкриття клієнт може записувати дані у поштову скриньку за допомогою функції *WriteFile* з такими параметрами:

hFile – описувач, що повертається функцією *CreateFile*.

lpBuffer – вказівник на буфер відправлення.

nNumberOfBytesToWrite – визначає, скільки байтів буде відправлено від клієнта до сервера (максимально – 64 Кбайти). Якщо описувач поштової скриньки створений з іменем домену чи з зірочкою, розмір повідомлення не повинен перевищувати 424 байти.

lpNumberOfBytesWritten – вказівник на змінну, в яку функція повертає кількість байтів, відправлених серверу.

lpOverlapped – повинен дорівнювати NULL. Функція WriteFile не блокує введення-виведення.

Лістинг 1.2 – Приклад клієнта поштової скриньки

```
// Client.cpp
#include <windows.h>
#include <stdio.h>
void main(int argc, char *argv[])
{
    HANDLE Mailslot;
    DWORD BytesWritten;
    char ServerName[256];
    // Перевірка наявності у командному рядку імені сервера
    if (argc < 2)
    {
        printf("Input server name:");
        scanf("\\\\%s\\Mailslot\\Myslot",ServerName);
    }
    else
        sprintf(ServerName,"\\\\%s\\Mailslot\\Myslot", argv[1]);
    if ((Mailslot = CreateFile(ServerName, GENERIC_WRITE,
    FILE_SHARE_READ, NULL, OPEN_EXISTING,
    FILE_ATTRIBUTE_NORMAL,
    NULL)) == INVALID_HANDLE_VALUE)
```

```
    {
        printf("CreateFile failed with error %d\n", GetLastError());
        return;
    }
    if (WriteFile(Mailslot, "This is a text from client ", 14, &BytesWritten,
NULL) == 0)
    {
        printf("WriteFile failed with error %d\n", GetLastError());
        return;
    }
    printf("Wrote %d bytes\n", BytesWritten);
    CloseHandle(Mailslot);
}
```

1.2 Іменовані канали

Іменовані канали – це простий механізм для забезпечення одностороннього або двостороннього передавання даних між процесами, оснований на архітектурі клієнт-сервер, який дозволяє скористатися вбудованими можливостями захисту Windows. Сервер може створити іменований канал і прийняти з'єднання з клієнтом. Клієнт ініціює з'єднання з існуючим сервером. Для обміну даними сервер і клієнт викликають функції ReadFile() і WriteFile(). Іменовані канали використовують два режими передавання даних – побайтовий режим і режим повідомлень.

Імена каналів мають такий формат

\\server\Pipe\ [шлях]ім'я,

де \\сервер – ім'я сервера, \Pipe – фіксований обов'язковий рядок, [шлях]ім'я – унікальне ім'я каналу. Ім'я сервера може бути подане крапкою.

У програмі, яка використовує іменований канал потрібно під’єднати заголовний файл Winbase.h. або Windows.h.

Сервер створює екземпляр іменованого каналу і отримує його описувач, який використовує для встановлення з'єднання з клієнтами і обміну з ними даними. Під час роботи сервер послідовно викликає такі функції:

- *CreateNamedPipe* – для створення іменованого каналу;
- *ConnectNamedPipe* – для прослуховування клієнтських з'єднань;
- *ReadFile* і *WriteFile* – для отримання і відправлення даних;
- *DisconnectNamedPipe* – для завершення з'єднання;
- *CloseHandle* – для закриття описувача іменованого каналу.

Створення іменованого каналу відбувається за допомогою API-функції ***CreateNamedPipe***:

```
HANDLE CreateNamedPipe(
LPCTSTR lpName,
DWORD dwOpenMode,
```

DWORD dwPipeMode,
DWORD nMaxInstances,
DWORD nOutBufferSize,
DWORD nInBufferSize,
DWORD nDefaultTimeOut,
LPSECURITY_ATTRIBUTES lpSecurityAttributes);

lpName – назва іменованого каналу. Це повинно бути ім’я локального комп’ютера або крапка.

dwOpenMode – напрямок передавання, керування введенням-виведенням і безпека каналу. У табл. 1.1 описані прапорці, комбінації яких використовують при створенні каналу.

Таблиця 1.1 – Прапорці режимів створення іменованого каналу

Режим відкриття	Прапорець	Опис
Напрямок	PIPE_ACCESS_DUPLEX	Канал двонаправлений: серверні і клієнтські процеси можуть приймати і відправляти дані через канал
	PIPE_ACCESS_OUTBOUND	Дані передаються тільки від сервера до клієнта
	PIPE_ACCESS_INBOUND	Дані передаються тільки від клієнта до сервера
Керування введенням-виведенням	FILE_FLAG_WRITE_THROUGH	Функції записування не повертають значення, доки дані передаються мережею або знаходяться у буфері віддаленого комп'ютера. Застосовується тільки у побайтовому режимі
	FILE_FLAG_OVERLAPPED	Дозволяє використовувати перекрите введення-виведення при виконанні операцій читання, записування і з'єднання
Безпека	WRITE_DACL	Дозволяє програмі змінювати список DACL іменованого каналу
	ACCESS_SYSTEM_SECURITY	Дозволяє програмі змінювати список SACL іменованого каналу
	WRITE_OWNER	Дозволяє програмі змінювати власника іменованого каналу і групового SID

dwPipeMode – визначає режими для операцій читання, записування і очікування. При створенні каналу потрібно вказати по одному прапорцю з кожної категорії параметрів, об'єднавши їх операцією OR, як показано у табл. 1.2.

nMaxInstances – визначає максимальну кількість каналів, що одночасно можуть бути створені сервером. Може приймати значення від 1 до PIPE_UNLIMITED_INSTANCES.

nOutBufferSize і *nInBufferSize* – розміри вхідного і вихідного буферів. Оптимальні розміри – ті, які програма використовує при виклику функцій *ReadFile* і *WriteFile*.

nDefaultTimeOut – час очікування з'єднання у мілісекундах. Параметр впливає тільки на клієнтські програми, які використовують функцію *WaitNamedPipe*.

lpSecurityAttributes – якщо цей параметр дорівнює NULL, іменований канал має ті ж самі права доступу, що і процес, який його створив.

Після створення каналу сервер очікує на з'єднання клієнтів за допомогою функції ***ConnectNamedPipe***, що визначена так:

BOOL ConnectNamedPipe(
HANDLE hNamedPipe,
LPOVERLAPPED lpOverlapped);

де *hNamedPipe* – описувач екземпляра іменованого каналу, повернутий функцією *CreateNamedPipe*.

lpOverlapped – дозволяє функції виконуватися асинхронно, якщо прапорець FILE_FLAG_OVERLAPPED був заданий при створенні каналу. Якщо даний параметр дорівнює NULL, функція *ConnectNamedPipe* блокується до тих пір, доки не встановиться з'єднання клієнта з сервером.

Таблиця 1.2 – Прапорці режимів читання-записування іменованого каналу

Режим	Прапорець	Опис
Записування	PIPE_TYPE_BYTE	Дані записуються у канал потоком байтів
	PIPE_TYPE_MESSAGE	Дані записуються у канал потоком повідомлень
Читання	PIPE_READMODE_BYTE	Дані зчитуються з каналу потоком байтів
	PIPE_READMODE_MESSAGE	Дані зчитуються з каналу потоком повідомлень
Очікування	PIPE_WAIT	Ввімкнено режим блокування
	PIPE_NOWAIT	Вимкнено режим блокування

Сервер відправляє і приймає дані за допомогою функцій *WriteFile* і *ReadFile*. Після завершення обміну сервер закриває з'єднання за допомогою функції *DisconnectNamedPipe*. У лістингу 1.3 подана програма сервера, що обмінюється даними з одним клієнтом.

Лістинг 1.3 – Сервер іменованого каналу

```
//Server.cpp
#include <windows.h>
#include <stdio.h>
void main(void)
{
```

```
    HANDLE PipeHandle;
    DWORD BytesRead;
    CHAR buffer[256];
    if ((PipeHandle = CreateNamedPipe("\\\\.\\Pipe\\Jim",PIPE_
    ACCESS_ DUPLEX, PIPE_TYPE_BYTE |PIPE_READMODE_BYTE, 1,
0, 0, 1000, NULL)) == INVALID_HANDLE_VALUE)
    {
        printf("CreateNamedPipe failed with error %d\n",GetLastError());
        return;
    }
    printf("Server is now running\n");
    if(ConnectNamedPipe(PipeHandle, NULL)==0)
    {
        printf("Error of ConnectNamedPipe ( %d)\n", GetLastError());
        CloseHandle(PipeHandle);
        return;
    }
    if(ReadFile(PipeHandle, buffer,sizeof(buffer),&BytesRead,NULL)<=0)
    {
        printf("ReadFile failed with error %d\n",GetLastError());
        CloseHandle(PipeHandle);
        return;
    }
    buffer[BytesRead]=0;
    puts(buffer);
    if(DisconnectNamedPipe(PipeHandle)==0)
    {
        printf("DisconnectNamedPipe failed with error %d\n",
        GetLastError());
        CloseHandle(PipeHandle);
        return;
    }
    CloseHandle(PipeHandle);
}
```

Клієнти можуть встановлювати з'єднання з існуючими на сервері каналами. Для створення простого клієнта потрібно виконати такі дії.

- Для перевірки наявності вільного екземпляра каналу викликати API-функцію *WaitNamedPipe*.
- Для встановлення з'єднання викликати API-функцію *CreateFile*.
- Для відправлення й одержання даних викликати функції *WriteFile* і *ReadFile*.
- Для завершення з'єднання викликати API-функцію *CloseHandle*.

Перед встановленням з'єднання клієнт повинен перевірити наявність на сервері вільного екземпляра іменованого каналу функцією ***WaitNamedPipe***:

BOOL WaitNamedPipe(
LPCTSTR lpNamedPipeName,
DWORD nTimeOut);

lpNamedPipeName – назва іменованого каналу.

nTimeOut – час очікування клієнтом вільного екземпляра каналу.

У випадку успішного завершення функції WaitNamedPipe потрібно відкрити екземпляр іменованого каналу за допомогою функції ***CreateFile***:

HANDLE CreateFile(
LPCTSTR lpFileName,
DWORD dwDesiredAccess,
DWORD dwShareMode,
LPSECURITY_ATTRIBUTES lpSecurityAttibutes,
DWORD dwCreationDisposition,
DWORD dwFlagsAndAttributes,
HANDLE hTemplateFile);

де *lpFileName* – назва каналу.

dwDesiredAccess – режим доступу, який повинен дорівнювати GENERIC_READ при читанні чи GENERIC_WRITE при записуванні у канал. Можна вказати обидва прапорці, об'єднавши їх за допомогою операції OR. Режим доступу повинен відповідати напрямку передавання (параметр *dwOpenMode*), заданому при створенні каналу.

Параметр *dwShareMode* повинен дорівнювати нулю, оскільки у кожен момент часу тільки один клієнт може одержати доступ до екземпляра каналу.

Параметр *lpSecurityAttibutes* повинен мати значення NULL, якщо не потрібно, щоб дочірній процес успадковував описувач клієнта.

dwCreationDispcistion – варто визначити як OPEN_EXISTING, тоді функція *CreateFile* буде повертати помилку, якщо каналу не існує.

dwFlagsAndAttributes – обов'язково повинен містити прапорець FILE_ATTRIBUTE_NORMAL. Крім того, можна вказати прапорці FILE_FLAG_WRITE_THROUGH, FILE_FLAG_OVERLAPPED і SECURITY_SQOS_PRESENT, об'єднавши їх логічною операцією OR.

hTemplateFile – повинен дорівнювати NULL.

Після успішного виконання функції CreateFile клієнт може відправляти і приймати дані за допомогою функцій ReadFile і WriteFile. Завершивши передавання, клієнт закриває з'єднання функцією CloseHandle.

У лістингу 1.4 наведено код клієнта іменованого каналу.

Лістинг 1.4 – Клієнт іменованого каналу

```
// Client.cpp
#include <windows.h>
#include <stdio.h>
```

```
#define PIPE_NAME "\\\\.\\Pipe\\Jim"
void main(void)
 {
HANDLE PipeHandle;
DWORD BytesWritten;
if (WaitNamedPipe(PIPE_NAME, NMPWAIT_WAIT_FOREVER) == 0)
{
    printf("WaitNamedPipe failed with error %d\n",GetLastError());
    return;
}
if ((PipeHandle = CreateFile(PIPE_NAME,GENERIC_READ |
GENERIC_WRITE, 0,(LPSECURITY_ATTRIBUTES) NULL,
OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, (HANDLE) NULL))
== INVALID_HANDLE_ VALUE)
{
    printf("CreateFile failed with error %d\n", GetLastError());
    return;
}
if (WriteFile(PipeHandle, "This is a test", 14, &BytesWritten, NULL)
== 0)
{
    printf("WriteFile failed with error %d\n", GetLastError());
    CloseHandle(PipeHandle);
    return;
}
printf("Wrote %d bytes", BytesWritten);
CloseHandle(PipeHandle);
}
```

1.3 Сокети windows

Winsock – це бібліотека, що являє собою інтерфейс мережевого програмування, незалежного від транспортного протоколу. Вона реалізована версіями Winsock 1.1 і Winsock 2.2. Програма, написана для Winsock 1.1, зможе виконуватись і з бібліотекою Winsock 2.2.

Перед викликом будь-якої функції Winsock необхідно завантажити правильну версію бібліотеки Winsock за допомогою функції ***WSAStartup***:

```
int WSAStartup(
WORD VersionRequested,
LPWSADATA lpWSAData);
```

Параметри функції.

VersionRequested – версія бібліотеки Winsock, яку необхідно завантажити. На сучасних платформах Windows використовується версія 2.2. Для завантаження версії Winsock 2.2 потрібно вказати значення 0x0202

або макрос MAKEWORD(2,2). Верхній байт визначає додатковий номер версії, нижній – основний.

lpWSAData – структура WSADATA, яка повертається після завершення виклику. Вона містить інформацію про версію Winsock, завантажену функцією *WSAStartup*.

Структура ***WSADATA*** описується так:

```
typedef struct WSAData
{
    WORD wVersion;
    WORD wHighVersion;
    char szDescription[WSADESCRIPTION_LEN +1];
    char szSystemStatus[WSASYS_STATUS_LEN+1];
    unsigned short iMaxSockets;
    unsigned short iMaxUdpDg;
    char * lpVendorlnfo;
} WSADATA, * LPWSADATA;
```

Єдина корисна інформація, що повертається у структурі WSADATA, – поля *wVersion* і *wHighVersion*.

Після завершення роботи з бібліотекою Winsock потрібно викликати функцію ***WSACleanup*** для вивантаження бібліотеки і звільнення ресурсів. Інтерфейс функції:

```
int WSACleanup (void);
```

Сокет – це описувач постачальника транспорту. Сокет створюється однією з двох функцій:

```
SOCKET socket (int af, int type, int protocol);
```

або

```
SOCKET WSASocket (
int af,
int type,
int protocol,
LPWSAPROTOCOL_INFO lpProtocolInfo,
GROUP g,
DWORD dwFlags);
```

Параметри функції такі:

af – визначає сім'ю адрес протоколу.

type – тип сокета для даного протоколу.

protocol – вказує конкретний транспорт.

У табл. 1.3 перераховані значення, використовувані у полях сім'ї адрес, типу сокета і протоколу для даного мережевого транспорту.

Найважливішим параметром сокета є сім'я адрес *af*. Цей параметр вказує протокол та обмежує значення інших параметрів. Сім'я адрес і тип сокета обмежують вибір транспортного протоколу. У такому разі можна задати у параметрі *protocol* значення нуль, а система сама обирає постачальника транспорту, виходячи з параметрів *af* і *type*. Але це можливо

лише для тих протоколів, для яких після виклику функції *lpProtocolInfo* – вказівник на структуру WSAPROTOCOL_INFO, яка повертається функцією *WSAEnumProtocols*.

g – завжди дорівнює нулю, оскільки Windows не підтримує групи сокетів.

dwFlags – може задаватись одним або декількома прапоцями:

WSA_FLAG_OVERLAPPED, WSA_MULTIPOINT_C_ROOT, WSA_MULTIPOINT_C_LEAF, WSA_MULTIPOINT_D_ROOT, WSA_MULTIPOINT_D_LEAF.

Таблиця 1.3 – Основні параметри сокетів

Протокол		Сокет		
Мережевий	Транспортний	Сім'я адрес (af)	Тип сокета (type)	Тип протоколу (protocol)
IP	TCP	AF_INET	SOCK_STREAM	IPPROTO_TCP
	UDP		SOCK_DGRAM	IPPROTO_UDP
	Прості сокети		SOCK_RAW	IPPROTO_RAW IPPROTO_IGMP

Перший прапорець – WSA_FLAG_OVERLAPPED, який встановлює перекрите введення-виведення, рекомендується задавади завжди при виклику функції *WSASocket*. Решта прапорців відносяться до багатоадресного передавання.

Щоб створити IP-сокет з використанням протоколу UDP, вказують тип сокета SOCK_DGRAM. Також можна створювати сокет для зв'язку безпосередньо через IP. Для цього задають тип сокета SOCK_RAW.

Основні виклики функцій, що повинні зробити клієнт і сервер для встановлення каналу зв'язку між комп'ютерами у мережі, зображені на рис. 1.1.

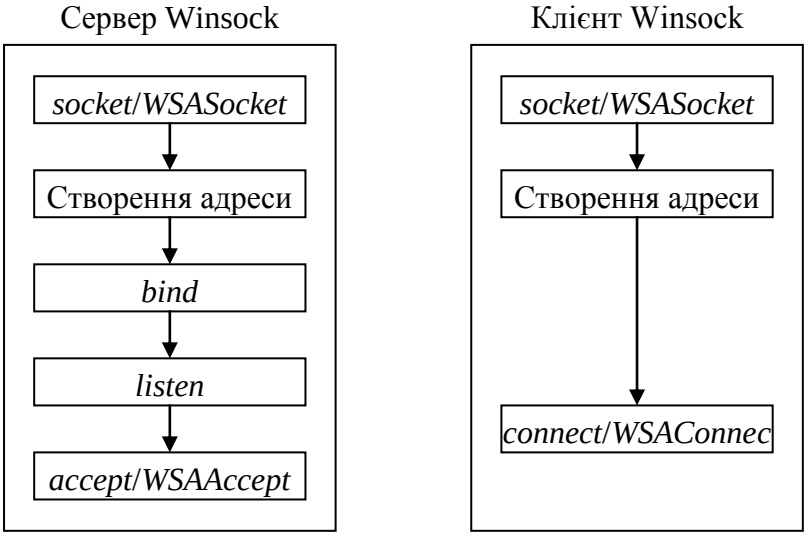


Рисунок 1.1 – Встановлення з'єднання

Розглянемо ці функції. Створення IP-сокета дозволяє програмам здійснювати під'єднання через TCP, UDP і протоколи IP. Для створення IP-сокета з використанням протоколу TCP викликаються функції *socket* чи *WSASocket* із сім'єю адрес AF_NET, типом сокета SOCK_STREAM, та значенням нуль поля протоколу, наприклад:

SOCKET s;

s=socket(AF_INET, SOCK_STREAM, 0);

s=WSASocket(AF_INET, SOCK_STREAM, 0, NULL, 0, WSA_FLAG_OVERLAPPED);

Для взаємодії із сервером через TCP чи UDP клієнт повинен вказати IP-адресу сервера і номер порту служби. Щоб прослуховувати вхідні запити клієнта, сервер також повинен вказати IP-адресу і номер порту. У Winsock IP-адреса і порт служби задають у структурі ***sockaddr_in***.

```
struct sockaddr_in
{
    short       sin_family;
    u_short     sin_port;
    in_addr     sin_addr;
    char        sin_zero[8];
};
```

Поля структури:

sin_family – повинно дорівнювати AF_INET;

sin_port – задає порт;

sin_addr – зберігає IP-адреси і є типом *unsigned long int*;

sin_zero – заповнювач, не задається.

IP-адреси звичайно задають у крапковій нотації a.b.c.d у вигляді рядка символів. Функція ***inet_addr*** перетворює IP-адресу з крапкової нотації у 32-бітове довге ціле без знака:

*unsigned long inet_addr(const char *cp);*

Параметр *cp* є рядком, що закінчується нульовим символом. Ця функція як результат повертає IP-адресу, подану 32-бітовим числом з мережевим порядком проходження байтів (*network-byte order*).

Спеціальні адреси. INADDR_ANY – дозволяє серверній програмі слухати з'єднаннння від будь-яких клієнтів. INADDR_BROADCAST – дозволяє широкомовно розсилати UDP-дайтаграми. Для її використання необхідно у програмі задати параметр сокета SO_BROADCAST. INADDR_LOOPBACK – адреса локального комп'ютера.

При використанні TCP і UDP програма вказує, через який порт зв'язатися.Номери портів поділяють на три категорії:

- 0–1023 – стандартні, контролюються IANA і зарезервовані для стандартних служб;
- 1024–49151 – зареєстровані IANA і можуть використовуватися процесами і програмами;
- 49152–65535 – динамічні (приватні).

IP-адреса та номер порту передаються у мережевому порядку (*network-byte-order*). У пам’яті комп’ютера числа зберігаються у системному порядку (*host-byte-order*). Чотири функції перетворяють числа із системного порядку у мережевий:

u_long htonl(u_long hostlong);

*int WSAHtonl (SOCKET s, u_longl hostlong, u_long * lpnetlong);*

u_short htons(u_short hostshort);

*int WSAHtons(SOCKET s, u_short hostshort, u_short * lpnetshort);*

де *hostlong* – числа типу long із системним порядком;

hostshort – числа типу *short* із системним порядком.

Чотири функції перетворяють числа із мережевого порядку у системний:

u_long ntohl(u_long netlong);

*int WSANtohl (SOCKET s, u_long netlong, u_long * lphostlong);*

u_short ntohs(u_short netshort);

*int WSANtohs(SOCKET s, u_short netshort, u_short * lphostshort);*

Функція *bind*. Зв'язує сокет із заданою адресою. Інтерфейс функції:

*int bind (SOCKET s, const sockaddr *name, int namelen);*

де *s* – сокет, на якому будуть очікуватись запити клієнтів на з'єднання;

name – вказівник на структуру, у якій знаходиться локальна адреса. При виклику *bind* потрібно привести цю структуру до типу *sockaddr**;

namelen – розмір передаваної структури адреси.

Функція *listen*. Переводить сокет у режим прослуховування. Інтерфейс функції:

int listen(SOCKET s, int backlog);

Параметри функції:

s – зв'язаний сокет;

backlog – максимальна довжина черги з'єднань, що очікують оброблення.

Приєднання нового сокета до прослуховувального сокета виконується за допомогою функцій *accept* і *WSAAccept*. Функція *accept*:

*SOCKET accept(SOCKET s, sockaddr *addr, int *addrlen);*

Параметри функції:

s – зв'язаний сокет у стані прослуховування;

addr – вказівник на адресу існуючої структури *sockaddr_in*;

addrlen – вказівник на довжину структури *sockaddr_in*.

Функція ***WSAAccept*** здатна встановлювати з'єднання залежно від результату обчислення умови:

SOCKET WSAAccept(
SOCKET s,
*sockaddr * addr,*
LPINT addrlen,
LPCONDITIONPROC lpfnCondition,
DWORD dwCallbackData);

Перші три параметри – ті ж, що і в *accept* для Winsock 1.

lpfnCondition – вказівник на функцію, що викликається при запиті клієнта. Вона визначає можливість приймання з'єднання і має такий прототип:

int CALLBACK ConditionFunc(
LPWSABUF lpCallerId.
LPWSABUF lpCallerData,
LPQOS lpSQOS.
LPQOS lpGQOS,
LPWSABUF lpCalleeId,
LPWSABUF lpCalleeData,
*GROUP * g.*
DWORD dwCallbackData);

lpCallerId – містить адресу об'єкта, що з'єднується.

Структура WSABUF використовується багатьма функціями Wipsock 2 і визначена так:

```
typedef struct __WSABUF
{
    u_long len;
    char * buf;
} WSABUF,* LPWSABUF;
```

де *len* – розмір буфера *buf* або кількість даних у ньому.

lpCallerId – вказівник на структуру WSABUF, поле *buf* якої вказує на структуру *sockaddr* з адресою віддаленого клієнта. Для доступу до інформації потрібно привести вказівник *buf* до типу *sockaddr_in*.

lpCallerData – не підтримується і дорівнює NULL.

lpSQOS – задає рівень якості обслуговування, запитуваний клієнтом. Посилається на структуру, що містить інформацію про вимоги пропускної здатності для приймання і передавання. Якщо клієнт не запитує параметри якості обслуговування (*quality of service, QoS*), то параметру задають значення NULL.

lpGQOS – не підтримується і дорівнює NULL.

lpCalleeId – вказівник на структуру WSABUF, поле *buf* якої вказує на структуру *sockaddr* з локальною адресою сервера.

lpCalleeData – не підтримується і дорівнює NULL.

Якщо з'єднання приймається, функція *ConditionFunc* поверне значення CF_ACCEPT, якщо відхиляється – CF_REJECT. Якщо на даний момент рішення не може бути прийнято, повертається CF_DEFER. Коли сервер готовий обробити запит, він викликає функцію *WSAAccept*.

Клієнтська частина значно простіша і для встановлення з'єднання потрібно всього три етапи: створити сокет функцією *socket* чи *WSASocket*, перетворити ім'я сервера (залежить від використовуваного протоколу) та ініціювати з'єднання функцією *connect* чи *WSAGonnect*.

Функція *connect*:

*int connect(SOCKET s, const sockaddr * name, int namelen);*

Параметри:

s – існуючий TCP-сокет для встановлення з'єднання;

name – структура, що містить адресу і порт сокета віддаленого сервера;

namelen – довжина змінної *name*.

Функція *WSAConnect*:

int WSAConnect(
SOCKET s,
*const sockaddr * name,*
int namelen,
LPWSABUF lpCallerData,
LPWSABUF lpCalleeData,
LPQOS lpSQOS,
LPQOS lpGQOS);

Перші три параметри такі ж, як і у функції *connect*.

lpCallerData – не підтримується і дорівнює NULL.;

lpCalleeData – не підтримується і дорівнює NULL..

Обидві змінні є структурами WSABUF, і для *lpCallerData* поле *len* повинно вказувати довжину буфера *buf* для передавання даних. У випадку *lpCalleeData* поле *len* визначає розмір буфера *buf* для приймання даних.

lpSQOS – вказівник на структуру *QoS*, що визначає вимоги до якості обслуговування, тобто до пропускної спроможності відправлення і приймання даних. Нульове значення параметра *lpSQOS* означає, що програма не висуває вимог до якості обслуговування.

lpGQOS – не підтримується і дорівнює NULL.

Для пересилання даних через сокет використовуються функції send та WSASend. Аналогічно, для приймання даних існують функцій recv і WSARecv.

Функція *send* визначена так

*int send(SOCKET S, const char * buf, int len, int flags);*

Параметри:

s – описувач сокета, через який відправляються дані;

buf – вказівник на символьний буфер з даними для відправлення;

len – містить кількість символів у буфері;

flags – встановлює режим передавання. Може набувати значень 0, MSG_DONTROUTE, MSG_OOB чи результату логічного АБО над двома останніми параметрами. При вказанні прапорця MSG_DONTROUTE транспорти не будуть маршрутизувати пакети, що відправляються. Прапорець MSG_OOB вказує, що дані повинні бути відправлені поза смугою (*out of band*), тобто терміново. Обробка цього запиту залишається на розсуд базового протоколу (наприклад, якщо транспорт не підтримує цей параметр, запит ігнорується).

Функція *WSASend* визначена так:

int WSASend (

SOCKET s,
LPWSABUF lpBuffers,
DWORD dwBufferCount,
LPDWORD lpNumberOfBytesSent,
DWORD dwFlags,
LPWSAOVERLAPPED lpOverlapped,
LPWSAOVERLAPPED_COMPLETION_ROUTINE lpCompletionRoutine);

Параметри:

s – описувач сокета для відправлення даних;

lpBuffers – вказівник на структуру WSABUF чи на масив цих структур;

dwBufferCount – кількість переданих структур WSABUF. Слід пам'ятати, що структура WSABUF містить сам символьний буфер і його довжину. Це називається комплексним введенням-виведенням (*scatter-gather I/O*). При використанні декількох буферів для відправлення даних через сокет з'єднання масив буферів відправляється, починаючи з першої і закінчуючи останньою структурою WSABUF;

lpNumberOfBytesSent – вказівник на тип DWORD, що після виклику *WSASend* містить загальне число переданих байтів;

dwFlags – такий же, що й у функції *send*;

1pOverlapped і *lpCompletionRoutine* – використовуються для перекритого введення-виведення (*overlapped I/O*) – однієї з моделей асинхронного введення-виведення, підтримуваних Winsock.

WSASend присвоює параметру *lpNumberOfBytesSent* кількість записаних байтів.

Функція *recv* визначена так:

*int recv(SOCKET s, char * buf, int len, int flags);*

Параметри:

s – сокет для приймання даних;

buf – символьний буфер, призначений для отриманих даних;

len – кількість прийнятих байтів чи розмір буфера *buf*;

flags – може приймати значення 0, MSG_PEEK, MSG_OOB чи результат логічного АБО над двома останніми параметрами. Нуль означає відсутність дій. MSG_PEEK вказує, що доступні дані повинні копіюватися у приймальний буфер і при цьому залишатися у системному буфері. Прапорець MSG_OOB вже обговорювався раніше при розгляді відправлення даних.

Функція *WSARecv* має додаткові, порівняно з *recv*, можливості: підтримує перекрите введення-виведення і фрагментарні дайтаграмні повідомлення.

int WSARecv(
SOCKET s,
LPWSABUF lpBuffers,
DWORD dwBufferCount,
LPDWORD lpNumberOfBytesRecvd,

LPDWORD lpFlags,
LPWSAOVERLAPPED lpOverlapped,
LPWSAOVERLAPPED_COMPLETION_ROUTINE lpCompletionRoutine);
Параметри:
s – сокет для приймання даних;
lpBuffers – вказівник на буфер для приймання даних;
dwBufferCount – довжина буфера;
lpBuffers – вказівник на масив структур WSABUF;
dwBufferCount – визначає кількість таких структур у масиві;
lpNumberOfBytesReceived – після одержання даних вказує на кількість отриманих байтів;

lpFlags – може набувати значення MSG_PEEK, MSG_OOB, MSG_PARTIAL чи результату логічного АБО над будь-якими з цих параметрів. MSG_PARTIAL використовується тільки з протоколами, орієнтованими на передавання повідомлень. Спочатку він встановлюється в нуль. При недостатньому розмірі вхідного буфера *WSARecv* кожен раз зчитує лише частину повідомлення і встановлює цей прапорець в одиницю, доки повідомлення не буде прочитане повністю. Якщо встановлений прапорець передається як вхідний параметр, операція приймання завершиться, навіть якщо це лише частина повідомлення.

lpOverlapped і *lpCompletionROUTINE* – застосовуються в операціях перекритого введення-виведення.

Після закінчення роботи із сокетом потрібно коректно завершити сеансза допомогою функції *shutdown*:

int shutdown(SOCKET s, int how);

Параметр *how* може набувати значення SD_RECEIVE, SD_SEND чи SD_BOTH. Значення SD_RECEIVE забороняє виклики функцій приймання даних. SD_SEND забороняє всі виклики функції відправлення даних. Нарешті, SD_BOTH забороняє як приймання, так і відправлення.

Після цього необхідно викликати функцію closesocket для закриття з'єднання і звільнення всіх ресурсів:

int closesocket (SOCKET s);

У протоколах, що не потребують з’єднання, архітектурою обміну є не клієнт-сервер, а приймач-передавач. Різниця між приймачем і передавачем полягає тільки у тому, хто викликає функції приймання, а хто – передавання інформації. Функція приймання завжди повинна викликатись першою. Будь-який з варіантів функції приймання виконується у блокувальному режимі і очікує отримання даних.

При побудові програми-приймача спочатку створюють сокет функцією socket чи WSASocket. Потім виконують прив'язку сокета до локальної адреси функцією bind (як і у випадку протоколів, орієнтованих на сеанси). Різниця у тому, не викликають функції listen та accept. Замість цього просто передають або отримують дані. Оскільки у цьому випадку

з'єднання немає, приймальний сокет може одержувати дайтаграми від будь-якої машини у мережі.

Найпростіша функція приймання – ***recvfrom***:

int recvfrom(
SOCKET s,
*char * buf,*
int len,
int flags,
*sockaddr * from,*
*int * fromlen);*

Перші чотири параметри такі ж, як і для функції recv, включно з допустимими значеннями для *flags*: 0, MSG_OOB і MSG_PEEK;

from – структура sockaddr передавального сокета, на розмір якої посилається fromlen.

Після повернення виклику структура *sockaddr* буде містити адресу робочої станції, що відправляє дані.

У Winsock 2 для приймання інформації без встановлення з'єднання застосовується ***WSARecvFrom***:

int WSARecvFrom(
SOCKET s,
LPWSABUF lpBuffers,
DWORD dwBufferCount,
LPDWORD lpNumberOfBytesRecvd,
LPDWORD lpFlags,
*sockaddr * lpFrom,*
LPINT lpFromlen,
LPWSAOVERLAPPED lpOverlapped,
LPWSAOVERLAPPED_COMPLETION_ROUTINE lpCompletionRoutine);

Функції можна надати один чи кілька буферів WSABUF, вказавши їх кількість у *dwBufferCount*, – у цьому випадку можливе комплексне введення-виведення. Сумарна кількість зчитаних байтів передається у *lpNumberOfBytesRecvd*. При виклику функції *WSARecvFrom lpFlags* може приймати такі значення: 0 (при відсутності параметрів), MSG_OOB, MSG_PEEK чи MSG_PARTIAL. Дані прапорці можна комбінувати логічною операцією АБО. Якщо при виклику функції задано прапорець MSG_PARTIAL, постачальник перешле дані навіть у випадку приймання лише частини повідомлення. Після повернення прапорець задається у MSG_PARTIAL, тільки якщо повідомлення прийняте частково. Після повернення *WSARecvFrom* присвоїть параметру *lpFrom* (вказівник на структуру *sockaddr*) адресу комп'ютера-відправника. Знову ж, параметр *lpFromLen* вказує на розмір структури SOCKADDR, однак у даній функції він є вказівником на DWORD. Два останніх параметри – *lpOverlapped* і *lpCompletionRoutine* – використовуються для перекритого введення-виведення.

Інший спосіб приймання-відправлення даних у сокетах, що не потребують з'єднання, – встановлення з'єднання (хоч це і звучить дивно). Після створення сокета і зв’язування з локальною адресою можна викликати connect чи *WSAConnect*, присвоївши параметру *sockaddr* адресу віддаленого комп'ютера, з яким необхідно зв'язатися. Фактично ніякого з'єднання не відбувається. Адреса сокета, переданого у функцію з'єднання, асоціюється із сокетом, щоб було можна використовувати функції *recv* і *WSARecv* замість *recvfrom* чи *WSARecvFrom* (оскільки джерело даних відоме). Якщо програмі потрібно одночасно зв'язуватися лише з однією кінцевою точкою, використовується можливість під’єднання сокета дайтаграм.

При побудові передавача є два способи відправлення даних через сокет, для яких не потрібне з’єднання. Перший і найпростіший – створити сокет і викликати функцію *sendto* чи *WSASendTo*.

Розглянемо функцію ***sendto***:

int sendto(
SOCKET s,
*const char * buf,*
int len,
int flags,
*const sockaddr * to,*
int tolen);

Параметри цієї функції такі ж, як і у *recvfrom*, за винятком *buf* – буфера даних для відправлення і *tolen* – розміра структури *sockaddr*.

Параметр *to* – вказівник на структуру *sockaddr* з адресою приймальної робочої станції.

Також можна використовувати функцію ***WSASendTo*** з Winsock 2:

int WSASendTo(
SOCKET s,
LPWSABUF lpBuffers,
DWORD dwBufferCount,
LPDWORD lpNumberOfBytesSent,
DWORD dwFlags,
*const sockaddr * lpTo,*
int iToLen,
LPWSAOVERLAPPED lpOverlapped,
LPWSAOVERLAPPED_COMPLETION_ROUTINE lpCompletionRoutine);

Функція *WSASendTo* аналогічна своїй попередниці. Вона приймає вказівник на одну чи кілька структур WSABUF з даними для відправлення одержувачу у вигляді параметра *lpBuffers*, а *dwBufferCount* задає кількість структур.

Для комплексного введення-виведення можна відправити кілька структур WSABUF. Перед виходом *WSASendTo* присвоює четвертому

параметру – *lpNumberOfBytesSent* – кількість реально відправлених одержувачу байтів. Параметр *lpTo* – структура *sockaddr* для даного протоколу з адресою приймача. Параметр *iToLen* – довжина структури *sockaddr*. Два останніх параметри – *lpOverlapped* і *lpCompletionRoutine* – використовуються для перекритого введення-виведення.

Як і при отриманні даних, сокет, що не потребує з'єднання, можна підключати до адреси кінцевої точки і відправляти дані функціями *send* і *WSASend*. Після створення цього прив'язування не можна використовувати *sendto* і *WSASendTo* з іншою адресою – буде видано помилку WSAEISCONN.

Протоколи, що потребують з'єднання, орієнтовані на передавання повідомлень. Тому потрібно враховувати такі фактори.

По-перше, оскільки орієнтовані на передавання повідомлень протоколи зберігають межі повідомлень, то дані, поставлені у чергу для відправлення, блокуються до завершення виконання функції відправлення. Якщо відправлення не може бути завершене, то при асинхронному або неблокувальному режимі введення-виведення функція відправлення поверне помилку WSAEWOULDBLOCK. Це означає, що операційна система не змогла обробити дані і потрібно викликати функцію відправлення повторно. Важливим є те, що в орієнтованих на повідомлення протоколах запис відбувається тільки внаслідок самостійної дії.

По-друге, при виклику функції приймання потрібно надати місткий буфер, інакше функція видасть помилку WSAEMSGSIZE: буфер заповнений, і дані, що залишилися, відкидаються. Винятком є протоколи, що підтримують обмін фрагментарними повідомленнями, наприклад AppleTalk PAP. Якщо була прийнята лише частина повідомлення, функція *WSARecvEx* присвоює параметру *flag* значення MSG_PARTIAL.

Для передавання дайтаграм на основі протоколів, що підтримують фрагментарні повідомлення, використовують одну з функцій *WSARecv* чи *recv*. Після чергового виклику *recv* може бути отримана лише частина дайтаграми. Проте при виклику *recv* не можна відстежити повноту зчитування повідомлення (це задача програміста). Через дане обмеження зручніше використовувати функцію *WSARecvEx*, що дозволяє задавати і зчитувати прапорець MSG_PARTIAL. Функції Winsock 2 *WSARecv* і *WSARecvFrom* також підтримують роботу з даним прапорцем.

Якщо замість *bind* викликається *sendto* чи *WSASendTo*, чи спочатку встановлюється з'єднання, мережевий стек автоматично вибирає локальну IP-адресу з таблиці маршрутизації. Таким чином, якщо спочатку була виконана прив'язка, початкова IP-адреса може бути неправильною і не відповідати інтерфейсу, з якого фактично була відправлена дайтаграма.

2 ЗАВДАННЯ ДО КУРСОВОЇ РОБОТИ

2.1 Загальні вимоги

Механізми обміну у конфігураціях

1 – Поштова скринька (Mailslot).

2 – Іменований канал (Namedpipe).

3 – Сокет із встановленням зв’язку (Winsock).

4 – Сокет без встановлення зв’язку (Winsock).

Отримання даних у програмах

A – масив чисел вводиться з клавіатури.

B – масив чисел зчитується з файлу.

C – перша частина масиву чисел вводиться з клавіатури, а друга частина зчитується з файлу.

D – частина масиву чисел зчитується з одного файлу, а частина з іншого.

E – перша частина масиву чисел зчитується з файлу, а друга частина вводиться з клавіатури.

Виведення результатів програмами

A – дані виводяться на екран і у файл.

B – дані вивводяться у файл.

C – дані виводяться на екран.

D – дані виводяться на екран.

E – дані виводяться на екран.

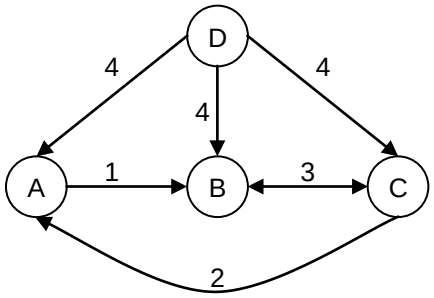
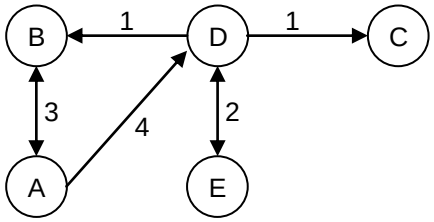
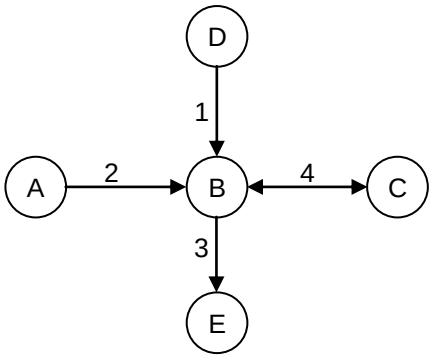
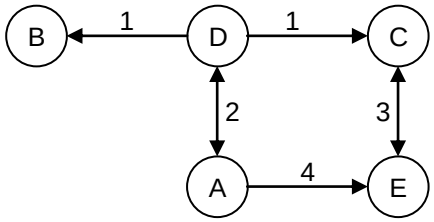
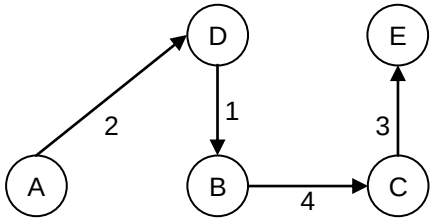
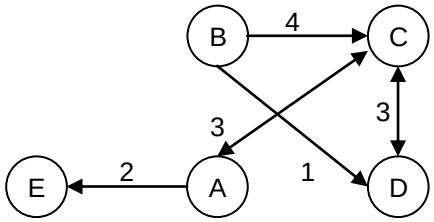
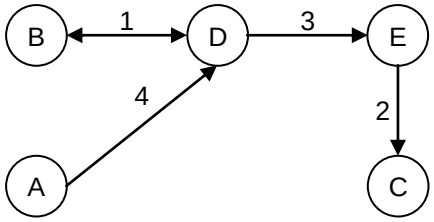
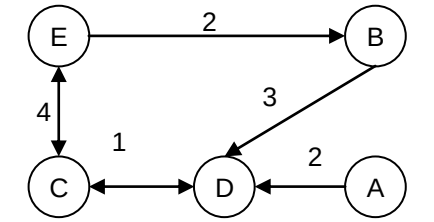
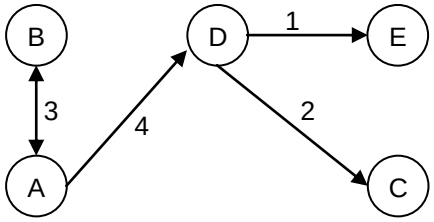
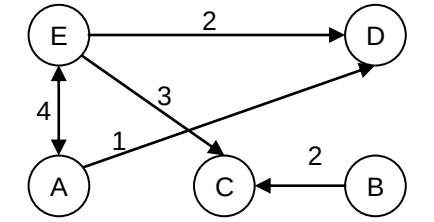
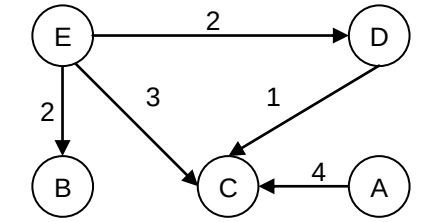
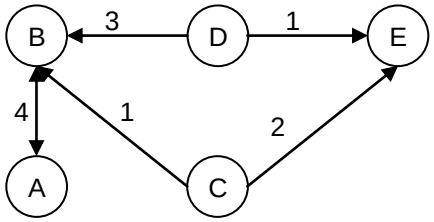
Формати даних та їх обробляння у програмах

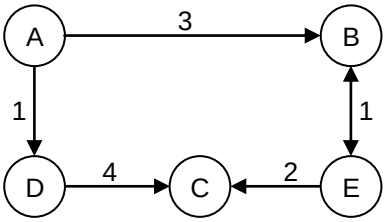
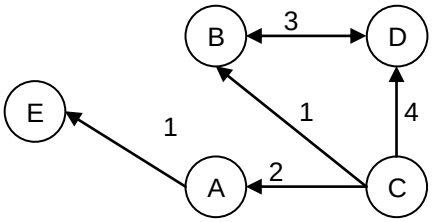
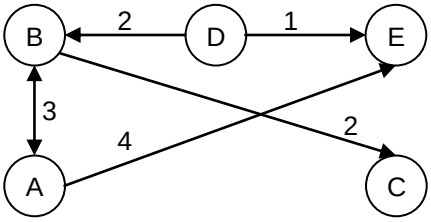
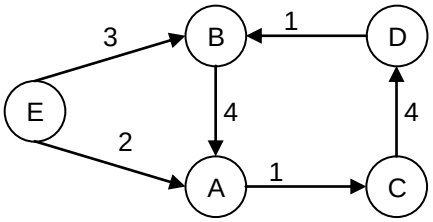
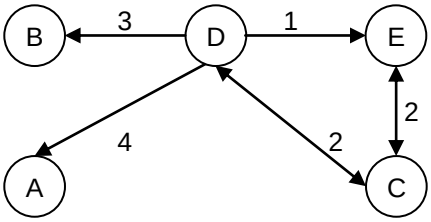
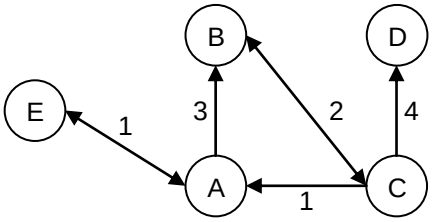
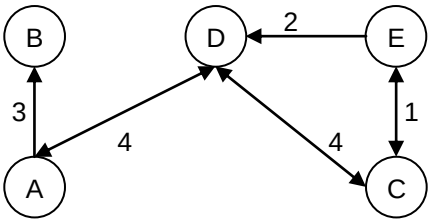
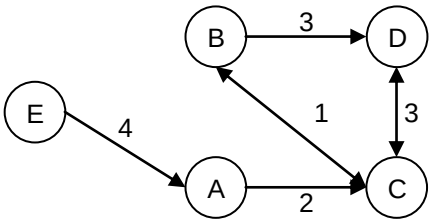
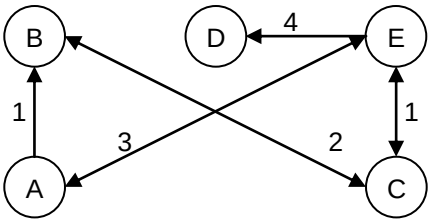
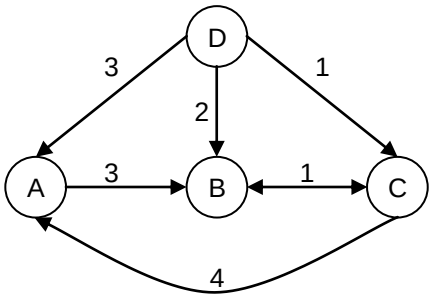
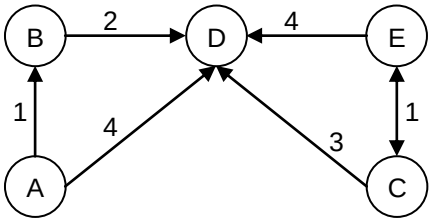
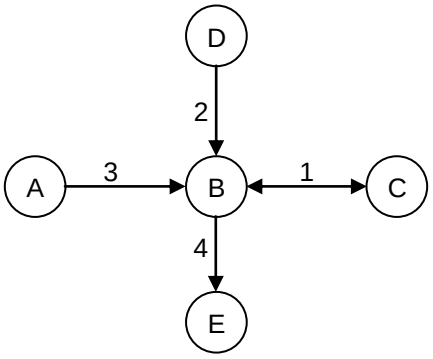
Всі дані є масивами одного типу. Якщо дані вводяться з клавіатури, то спочатку вводиться їх кількість. Кількість даних у файлах повинна визначатись у програмі. Для зберігання вхідних і вихідних даних потрібно використовувати динамічно створені масиви. Програми повинні формувати масиви вихідних даних на основі обробляння вхідних масивів введених з клавіатури, зчитаних з файлу та отриманих від інших програм. Слід враховувати, що довжини масивів даних в програмі можуть бути різними.

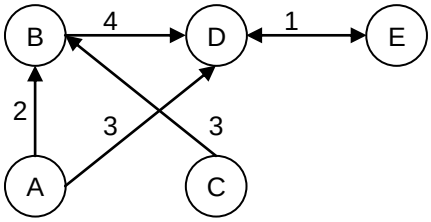
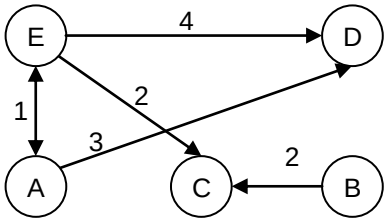
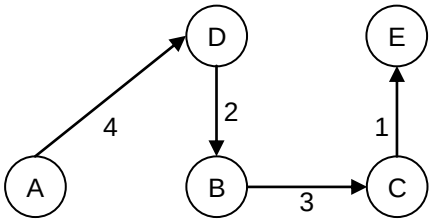
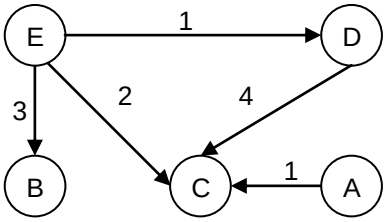
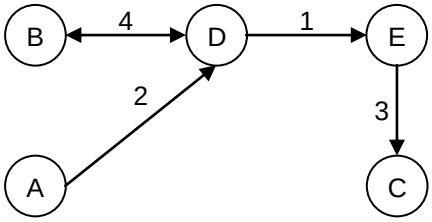
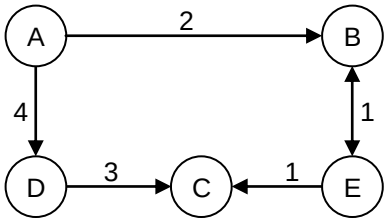
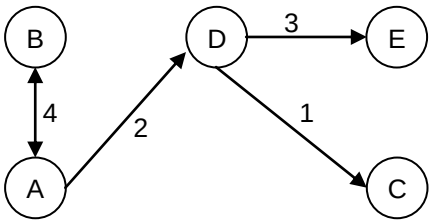
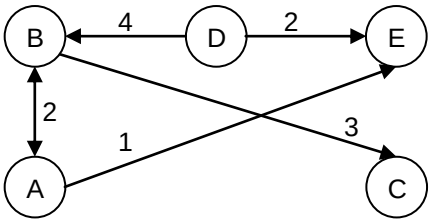
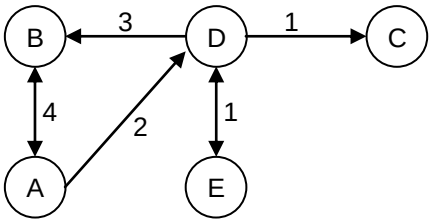
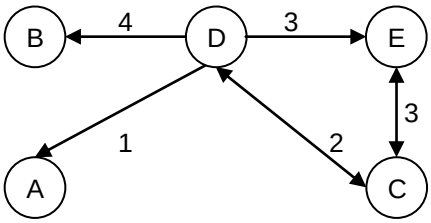
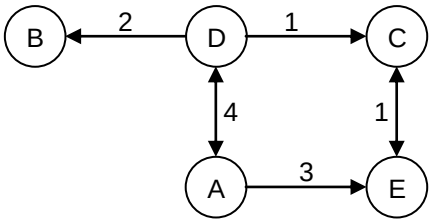
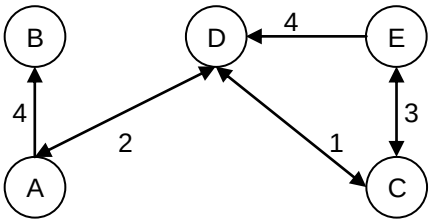
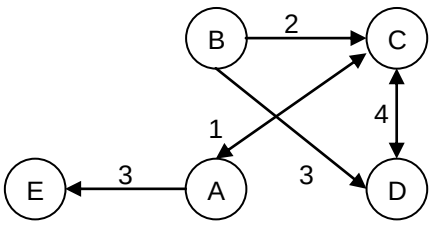
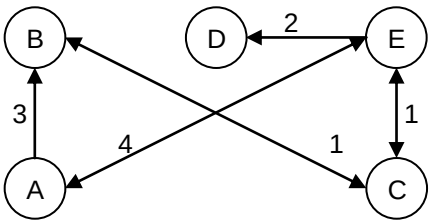
2.2 Варіанти завдань

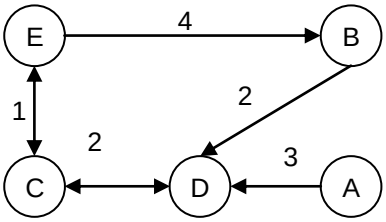
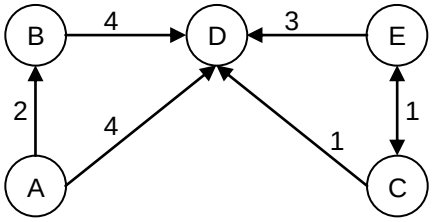
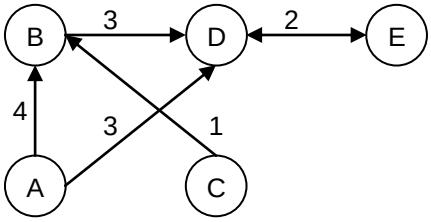
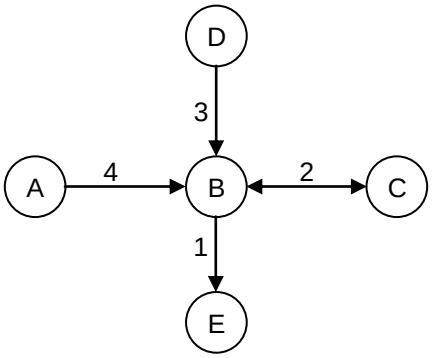
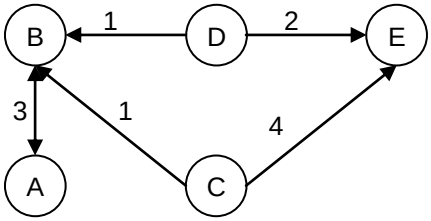
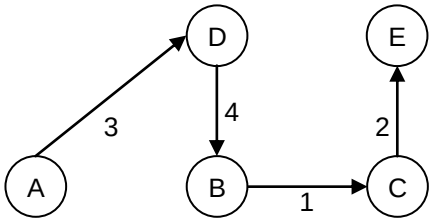
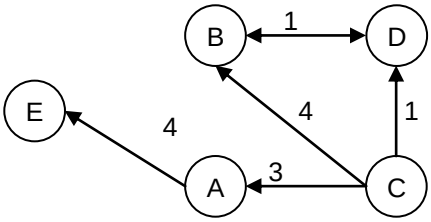
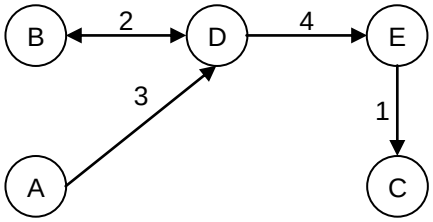
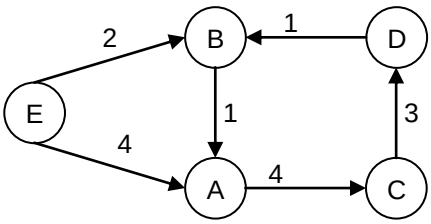
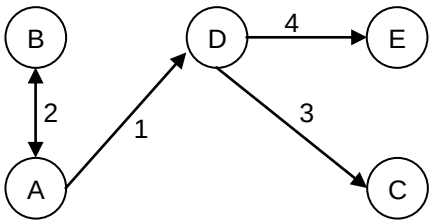
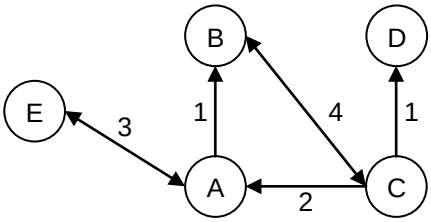
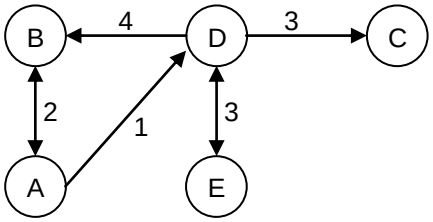
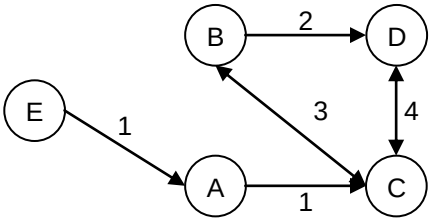
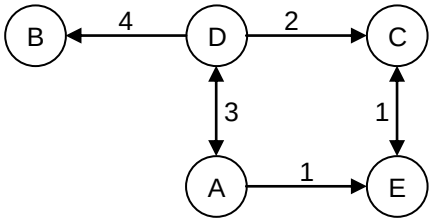
У таблиці 2.1 надано варіанти конфігурацій обміну інформацією відповідно до номерів завдань на курсову роботу.

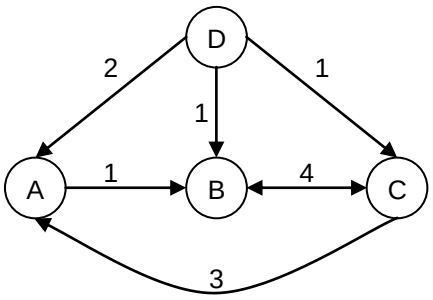
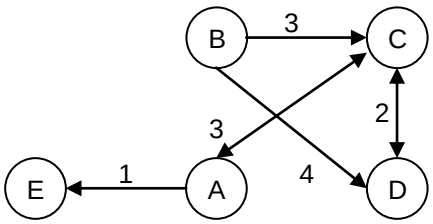
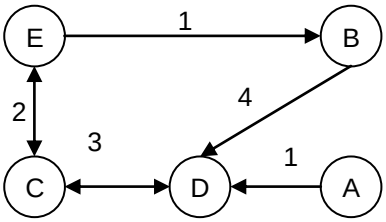
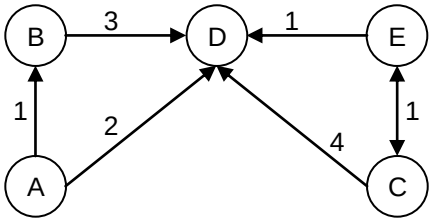
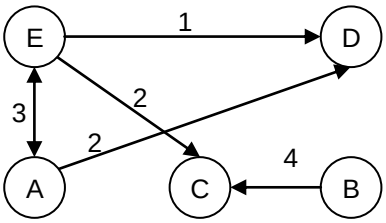
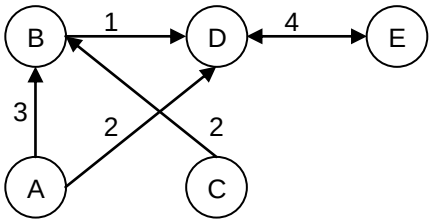
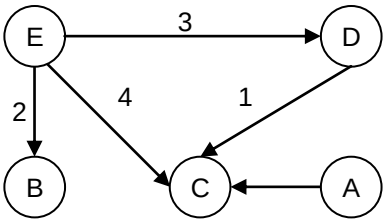
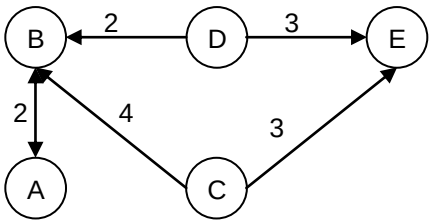
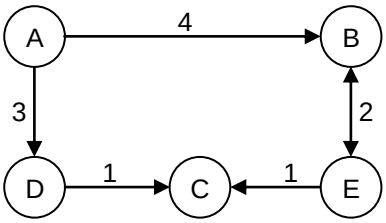
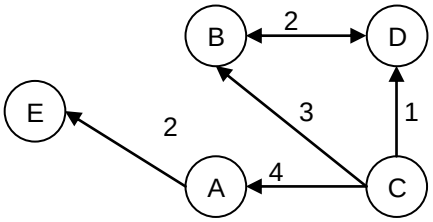
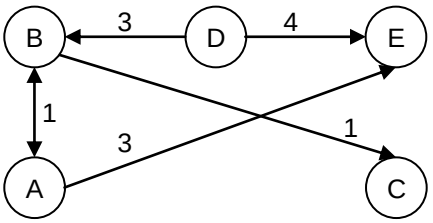
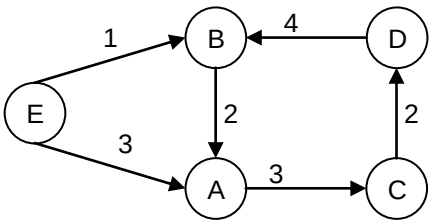
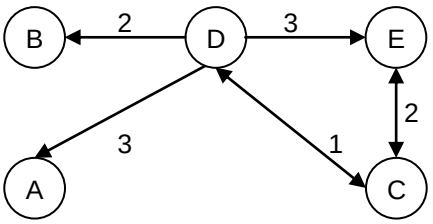
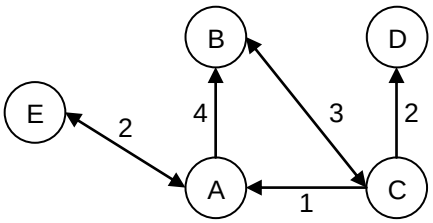
Таблиця 2.1 – Варіанти конфігурацій обміну

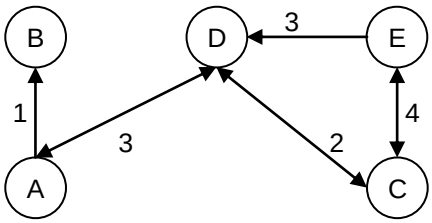
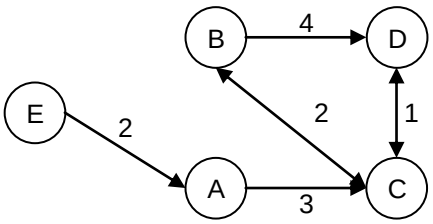
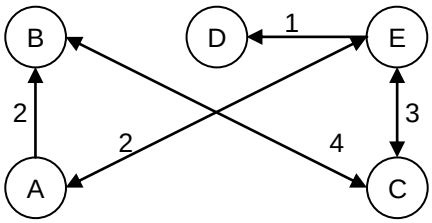
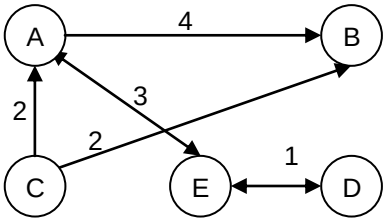
№	Конфігурація	№	Конфігурація
1		2	
3		4	
5		6	
7		8	
9		10	
11		12	

№	Конфігурація	№	Конфігурація
13		14	
15		16	
17		18	
19		20	
21		22	
23		24	

№	Конфігурація	№	Конфігурація
25		26	
27		28	
29		30	
31		32	
33		34	
35		36	
37		38	

№	Конфігурація	№	Конфігурація
39		40	
41		42	
43		44	
45		46	
47		48	
49		50	
51		52	

№	Конфігурація	№	Конфігурація
53		54	
55		56	
57		58	
59		60	
61		62	
63		64	
65		66	

№	Конфігурація	№	Конфігурація
67		68	
69		70	

Завдання можуть змінюватись та доповнюватись викладачем.

3 РЕКОМЕНДАЦІЇ ЩОДО ОФОРМЛЕННЯ КУРСОВОЇ РОБОТИ

3.1 Правила оформлення пояснювальної записки

Пояснювальна записка повинна бути оформлена відповідно до вимог ДСТУ 3008–95, "Єдиної системи програмної документації (ЄСПД)" та "Методичних вказівок до оформлення курсових проектів (робіт) у Вінницькому національному технічному університеті". Текст пояснювальної записки повинен бути надрукований у редакторі Microsoft Word на стандартних аркушах формату А4. Обсяг пояснювальної записки не повинен перевищувати 25–30 сторінок. Шрифт основного тексту – Times New Roman, розмір 14 пт. Міжрядковий інтервал встановлювати – 1,5. Відступи від країв аркуша: верхнього – 25 мм, нижнього і лівого – 20 мм; правого – 10 мм. Абзац – 1,27мм. Нумерація сторінок ставиться у правому верхньому куту, починаючи зі змісту. Сторінки до змісту не нумеруються, але враховуються при нумерації. Заголовки структурних частин, розділів оформлюються великими літерами посередині рядка, всі інші з абзацу малими літерами починаючи з великої. Слово "Додаток" оформлюється малими літерами з першої великої посередині рядка. Запис літературного джерела: "ПЕРЕЛІК ПОСИЛАНЬ". У переліку посилань повинні бути ті літературні джерела, на які є посилання у тексті пояснювальної записки. Кожен розділ рекомендується починати з нової сторінки. Заголовок розділу записуюється посередині великими буквами з більш високою насиченістю. Заголовки підрозділів, пунктів та підпунктів (при наявності заголовка) записуються з абзацу малими буквами починаючи з великої. Розділи нумеруються порядковими номерами в межах всього документа (1, 2, і т. д.). Після номера крапка не ставиться, а пропускається один знак. Підрозділи нумеруються в межах кожного розділу, пункти в межах підрозділу і т. д. за формою (3.1, 3.2, 3.2.1, 3.2.2, 3.2.2.1 і т. д.). Цифри, які вказують номер, не повинні виступати за абзац. Посилання в тексті на розділи виконується за формою: "...наведено в розділі 3".

3.2 Структура пояснювальної записки

Курсова робота повинна мати таку структуру.

1. Титульна сторінка.

Оформлюється за зразком, наведеним у п. 2.3. Титульна сторінка не нумерується, але враховується при нумерації.

2. Індивідуальне завдання до курсової роботи.

Оформлюється за зразком, наведеним у п. 2.4. Підписується студентом, керівником та завідувачем кафедри. Індивідуальне завдання не нумерується, але враховується при нумерації.

3. Анотація.

Містить 5–7 речень, у яких наводиться короткий опис пояснювальної записки. Анотація не нумерується, але враховується при нумерації.

4. Зміст.

З даної сторінки починається нумерація, але сама сторінка у змісті не вказується. У змісті вказуються усі розділи і підрозділи, а також додатки у такому ж форматі, в якому вони подані у тексті пояснювальної записки.

5. Вступ.

Вступ висвітлює: стан розвитку проблеми в даній галузі, до якої має відношення розробка; галузь використання та призначення; мету та загальну постановку задачі; актуальність, яка повинна подаватись в останньому абзаці вступу. Кількість сторінок вступу не повинна перевищувати, 1–2 сторінок.

6. Аналіз сучасного стану мережевих технологій програмування та обгрунтування теми.

Цей розділ є обов'язковим та передбачає посилання до відомих літературних джерел, враховуючи тенденції розвитку та сучасний стан програмування. У цьому розділі необхідно порівняти основні відомі технології програмування та мови, що їх застосовують.

7. Розробка програмного забезпечення виконання завдання.

7.1. Розробка режимів роботи та обміну інформацією.

Описуються і пояснюються дії, які виконує кожна програма для отримання даних. Наводиться конфігурція обміну даних між програмами та описується його порядок. Можуть бути наведено узагальнені алгоритми роботи програм.

7.2. Структура даних і функцій.

Описуються всі змінні та функції програм. В описі змінних потрібно вказувати їх тип, модифікацію, призначення, область видимості та час існування. В описі функцій потрібно вказувати їх призначення, типи і призначення параметрів, тип значення, що повертається.

8. Розробка та виконання тестового прикладу.

Сюди входять: опис форматів даних, що вводяться з клавіатури, чи зчитуються з файлів, роздруковані тексти вихідних даних та одержаних результатів, роздруковані графічні результати виконання програми, а також зовнішній вигляд меню.

9. Інструкція користувача.

Містить: вимоги щодо апаратної частини та програмного забезпечення комп'ютера, на якому планується використовувати програмний продукт (процесор, об'єм пам'яті, відеокарта, тип операційної системи тощо); рекомендації щодо інсталяції та запуску програмного продукту; інструкція для роботи з програмою.

10. Висновки.

Подається коротка узагальнена характеристика одержаних результатів, їх відповідності індивідуальному завданню на курсову роботу, переваги і недоліки технології WinAPI.

11. Перелік посилань.

Наводяться літературні джерела, на які є посилання у тексті пояснювальної записки. Літературні джерела нумеруються у порядку посилань.

12. Додатки.

Перший рядок додатка у центрі повинен містити слово "додаток", після якого через пробіл пишеться номер додатка. Додатки нумеруються великими буквами українського алфавіту. Другий рядок повинен містити у центрі слово "(обов’язковий)" або "(довідковий)" у круглих дужках. Третій рядок повинен містити тематичний заголовок додатка. Першими обов’язковими додатками повинні бути роздруковані тексти програм, що виконують індивідуальне завдання. Можуть бути також довідкові додатки.

ПІСЛЯМОВА

У методичних вказівках надано інформацію, необхідну для виконання курсових робіт з дисципліни "Прикладне програмування".

Надано інформацію про принципи мережевого програмування з використанням поштових скриньок. Розглянуто переваги і недоліки даного підходу. Розглянуто використання технології іменованих каналів для мережевого програмування, яка надає просту архітектуру "клієнт-сервер" для надійного передавання даних з використанням перенаправлювача Windows. Основна перевага іменованих каналів полягає у тому, що вони дозволяють скористатися вбудованими можливостями захисту операційної системи Windows. Описано аспекти популярної технології мережевого програмування з використанням сокетів Windows. У рамках створення WinAPI-програм розглянуто технологію Winsock. Winsock – це інтерфейс прикладного програмування мережевих взаємодій, реалізований на всіх платформах Windows. У середовищі Windows даний інтерфейс став повністю незалежним від мережевих протоколів. Описано режими сокетів, алгоритм встановлення з'єднання та функції, необхідні для встановлення з'єднання і виконання обміну. При написанні теоретичних відомостей даних вказівок використано матеріали з [1, 2].

Наведено варіанти завдань для розробки мережевих прогам з використанням оисаних механізмів обміну інформацією. Кожне завдання потребує розробки декількох програм і має унікальну конфігурацію обміну інформацією між ними.

Описано вимоги і рекомендації щодо оформлення пояснювальної записки до курсової роботи, які сформовано на основі матеріалів [3, 4]. У додатках наведено приклади оформлення титульної сторінки та індивідуального завдання, а також надано список рекомендованої літератури.

Виконання курсової роботи передпачає використання бібліотеки програмування WinAPI, на якій базуються всі інші технології Microsoft. Однак, Microsoft продовжує розвивати технології мережевого програмування. У рамках програмного середовища Visual C++ запропоновано класи сокетів MFC. Бібліотека MFC є об'єктно-орієнтованою надбудовою над WinAPI, призначеною для спрощення програмування за рахунок використання класів, що інкапсулюють функції більш високого рівня. У MFC сокет – це об'єкт відповідного класу. Детальнішу інформацію про технологію програмування за допомогою сокетів MFC можна отримати у [5].

Останнім часом широкої популярності набула нова платформа програмування Microsoft, що отримала назву ".NET" (DOT NET). У рамках даної платформи спеціально для мережевого і розподіленого програмування розроблено мову C#, що є модифікацією мови C++. Тому для вивчення C# найкраще мати досвід з програмування на C++. Тим, хто

знайомий з C++, для вивчення мови C# можна порекомендувати [6]. У даній книзі описана розробка програм мовою C# для платформи DOT NET, а також аспекти мережевого програмування. Для більш детального вивчення технології програмування у мережах з використанням сокетів та інших засобів платформи DOT NET можна порекомендувати [7].

Дані методичні вказівки є основою для отримання приктичних навичок з вивчення дисципліни "Прикладне програмування". Проте вивчення цієї дисципліни потребує наявності базових знань як з програмного, так і з апаратного забезпечення засобів сучасної цифрової техніки. Для отримання базової підготовки з програмування існує велика кількість широко відомих літературних джерел. Для отримання знань з передових досягнень у галузі побудови засобів цифрової техніки можна порекомендувати наукові роботи [8-10].

ЛІТЕРАТУРА

1. Азаров О. Д. Прикладне програмування : навч. посібник / О. Д. Азаров, О. І. Черняк, Л. А. Савицька – Вінниця : ВНТУ, 2016. – 131 с.
2. Джонс Э. Программирование в сетях Microsoft Windows. Мастер-класс. / Э. Джонс, Дж. Оланд ; пер. с англ. – Спб. : Питер; М. : Издательско-торговый дом "Русская редакция", 2005. – 608 с. : ил.
3. ДСТУ 3008–95. "Документація. Звіти у сфері науки і техніки. Структура і правила оформлення".
4. Лисенко Г. Л. Методичні вказівки до оформлення курсових проектів (робіт) у Вінницькому національному технічному університеті / Уклад. Г. Л. Лисенко, А. Г. Буда, Р. Р. Обертюх. – Вінниця : ВНТУ, 2006. – 60 с.
5. Шеферд Джордж. Программирование на Microsoft Visual C++ .NET. / Д. Шеферд ; пер. с англ. – М. : Издательство-торговый дом "Русская Редауция", 2003. – 928 с. : ил.
6. Нэш Т. C# 2008: ускоренный курс для профессионалов. Т. Нэш ; пер. с англ. – М. : ООО "И. Д. Вильямс", 2008. – 576 с. : ил.
7. Кровчик Э. .NET. Сетевое программирование для профессионалов. / Э. Кровчик, В. Кумар, Н. Лагари и др. ; пер. с англ. – М. : Издательство "ЛОРИ", 2007. – 400 с. : ил.
8. Азаров О. Д. Повнофункціональна побітова потокова арифметика зі зменшеними витратами обладнання. : монографія / О. Д. Азаров, О. І. Черняк. – Вінниця : ВНТУ, 2013. – 200 с.
9. Азаров О. Д. Структурна організація побітового додавання і віднімання кодів золоттої 1-пропорції із врахуванням знаків / О. Д. Азаров, О. І. Черняк // Інформаційні технології та комп’ютерна інженерія. Вінницький національний технічний університет – 2011. – № 3(22). С. 13–16.
10. Азаров О. Д. Метод побудови швидкодіючих фібоначчієвих лічильників / О. Д. Азаров, О. І. Черняк // Проблеми інформатизації та управління – 2014. – № 2(46). – С 5–8.

ДОДАТОК А
(обов'язковий)
Зразок титульної сторінки

Міністерство освіти і науки України
Вінницький національний технічний університет
Кафедра обчислювальної техніки

КУРСОВА РОБОТА

з дисципліни "Прикладне програмування"
на тему

РОЗРОБКА ПРОГРАМ ОБМІНУ ІНФОРМАЦІЄЮ У МЕРЕЖАХ WINDOWS

Студента (ки) ___ курсу групи_____
спеціальності "Комп'ютерні системи і мережі"

_____
(прізвище та ініціали)

Керівник к.т.н., доц. Черняк О. І.

Національна шкала_____

Кількість балів_____ Оцінка ECTS_____

Члени комісії _____ _____
(підпис) *(прізвище та ініціали)*

_____ _____
(підпис) *(прізвище та ініціали)*

м. Вінниця 2016 рік

ДОДАТОК Б
(обов'язковий)
Зразок індивідуального завдання

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра обчислювальної техніки

ЗАТВЕРДЖУЮ
В. о. зав. кафедри ОТ, к.т.н., доц.
_____Л. В. Крупельницький
(підпис)
"___"_____ 201_ р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ
на курсову роботу з дисципліни "Прикладне програмування"

студенту _____

факультету _____групи_____

Тема: Розробка програм обміну інформацією у мережах Windows

Завдання № **17** *Номер варіанта завдання*

1 Постановка задачі

Розробити комплекс програм, що обмінюються між собою інформацією за допомогою мережевих технологій Microsoft. Комплекс складається з програм A, B, C, D, E, що утворюють конфігурацію, зображену на рисунку 1.

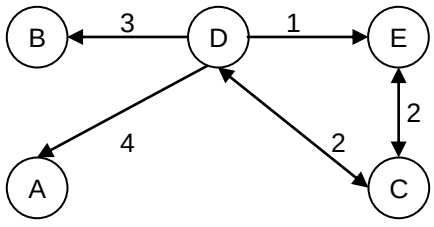


Рисунок 1 – Конфігурація програмного комплексу

1.1. Програми повинні формувати масиви результатів на основі обробляння масивів даних введених з клавіатури, зчитаних з файлу та отриманих від інших програм відповідно до вимог завдання.

1.2. Всі дані є масивами одного типу. Якщо дані вводяться з

клавіатури, то спочатку вводиться їх кількість. Кількість даних у файлах повинна визначатись у програмі.

1.3. Для зберігання вхідних і вихідних даних потрібно використовувати динамічно створені масиви. Слід враховувати, що довжини масивів даних в програмі можуть бути різними.

1.4. Програми повинні виводити результати на екран або у файл відповідно до вимог завдання.

2 Вимоги до завдання

2.1. Механізми обміну у конфігураціях:

1 – Поштова скринька (Mailslot).
2 – Іменований канал (Namedpipe).
3 – Сокет із встановленням зв’язку (Winsock).
4 – Сокет без встановлення зв’язку (Winsock).

2.2. Отримання даних у програмах:

A – масив чисел вводиться з клавіатури.
B – масив чисел зчитується з файлу.
C – перша частина масиву чисел вводиться з клавіатури, а друга частина зчитується з файлу.
D – частина масиву чисел зчитується з одного файлу, а частина з іншого.
E – перша частина масиву чисел зчитується з файлу, а друга частина вводиться з клавіатури.

2.3. Виведення результатів програмами

A – дані виводяться на екран і у файл.
B – дані вивводяться у файл.
C – дані виводяться на екран.
D – дані виводяться на екран.
E – дані виводяться на екран.

Дата видачі " ___" _____201_ р.

Керівник:
к.т.н., доц. кафедри ОТ _____О. І. Черняк
(підпис)

Завдання отримав _____ _____
(підпис) *(ПІБ)*

ДОДАТОК В
(обов’язковий)
Список рекомендованої літератури

1. ДСТУ 3008–95. "Документація. Звіти у сфері науки і техніки. Структура і правила оформлення".
2. Лисенко Г. Л. Методичні вказівки до оформлення курсових проектів (робіт) у Вінницькому національному технічному університеті / Уклад. Г. Л. Лисенко, А. Г. Буда, Р. Р. Обертюх. – Вінниця : ВНТУ, 2006. – 60 с.
3. Азаров О. Д. Методичні вказівки до виконання курсової роботи з дисципліни "Прикладне програмування" для студентів напряму підготовки – "Комп’ютерна інженерія" / Уклад. О. Д. Азаров, О. І. Черняк, Л. А. Савицька – Вінниця : ВНТУ, 2016 – 43 с.
4. Азаров О. Д. Прикладне програмування : навч. посібник / О. Д. Азаров, О. І. Черняк. – Вінниця : ВНТУ, 2016. – 131 с.
5. Джонс Э. Программирование в сетях Microsoft Windows. Мастер-класс. / Э. Джонс, Дж. Оланд ; пер. с англ. – Спб. : Питер; М. : Издательско-торговый дом "Русская редакция", 2005. – 608 с. : ил.
6. Грегори К. Использование Visual C++ 6. Специальное издание. / К. Грегори ; пер. с англ. – К. : Диалектика, 2000. – 816 с. : ил.
7. Снейдер Й. Эффективное программирование TCP/IP. Библиотека программиста. / Й. Снейдер ; пер. с англ. – СПб. : "Питер", 2001. – 320 с. : ил.
8. Нэш Т. C# 2008: ускоренный курс для профессионалов. Т. Нэш ; пер. с англ. – М. : ООО "И. Д. Вильямс", 2008. – 576 с. : ил.
9. Кровчик Э. .NET. Сетевое программирование для профессионалов. / Э. Кровчик, В. Кумар, Н. Лагари и др. ; пер. с англ. – М. : Издательство "ЛОРИ", 2007. – 400 с. : ил.

Навчальне видання

Методичні вказівки до виконання курсової роботи з дисципліни "Прикладне програмування" для студентів спеціальності "Комп'ютерна інженерія" всіх форм навчання

Редактор В. Дружиніна

Укладачі: Азаров Олексій Дмитрович
Черняк Олександр Іванович
Савицька Людмила Анатоліївна

Оригінал-макет підготовлено О. І. Черняком

Підписано до друку
Формат 29,7×42 ¼ . Папір офсетний
Гарнітура Times New Roman.
Друк різографічний. Ум. друк. арк.
Наклад пр. Зам. №

Вінницький національний технічний університет,
навчально-методичний відділ ВНТУ,
21021, м. Вінниця, Хмельницьке шосе, 95,
ВНТУ, к. 2201
Тел. (0432)59-87-36.
Свідоцтво суб'єкта видавничої справи
серія ДК № 3516 від 01.07.2009 р.

Віддруковано у Вінницькому національному технічному університеті
в комп'ютерному інформаційно-видавничому центрі.
21021, м. Вінниця, Хмельницьке шосе, 95,
ВНТУ, ГНК, к.114.
Тел. (0432)59-87-38.
publish.vntu.edu.ua; e-mail kivc.vntu@gmail.com
Свідоцтво суб'єкта видавничої справи
серія ДК № 3516 від 01.07.2009 р.