МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Одеська національна академія зв'язку ім. О. С. Попова

Кафедра інформаційних технологій

КОМП'ЮТЕРНІ ТЕХНОЛОГІЇ ТА ПРОГРАМУВАННЯ

Частина 3

ПРОГРАМНЕ ОПРАЦЮВАННЯ ФАЙЛІВ

Методичні вказівки до лабораторних і практичних робіт для студентів напряму 6.050202 "Автоматизація та комп'ютерно-інтегровані технології"

Одеса 2016

УДК 004.43 План НМВ 2016 р.

Укладачі: **Трофименко О.Г., Прокоп Ю.В., Буката Л.М.**

Рецензент: к.т.н., доцент Флейта Ю.В.

Методичні вказівки призначені для студентів при підготовці до лабораторних і практичних занять з дисципліни “Комп’ютерні технології та програмування”. Оскільки ця дисципліна передбачає значну кількість лабораторних і практичних занять, методичний матеріал для їх підготовки було поділено на декілька частин.

У третій частині розглянуто засоби Visual C++ програмного опрацювання різнотипних файлів: створення, заповнення даними, читання, редагування та ін. Саме цим темам, згідно з навчальною програмою, присвячені з двадцять першої по двадцять восьму лабораторні роботи курсу. Наведені роботи містять короткі теоретичні відомості, приклади створювання програмних проектів засобами Visual C++, питання для самоконтролю та варіанти індивідуальних завдань різних рівнів складності для виконання їх на комп’ютері.

Методичні вказівки будуть корисними студентам спеціальності “Автоматизація та комп’ютерно-інтегровані технології”, які вивчають дисципліну “Комп’ютерні технології та програмування” для набуття навиків програмування з метою подальшого застосовування цих навиків у власній повсякденній і майбутній професійній діяльності; також стануть у нагоді користувачам персональних комп’ютерів, які бажають навчитися програмувати у середовищі Visual C++.

СХВАЛЕНО
на засіданні кафедри
інформаційних технологій
і рекомендовано до друку.

Протокол № 4 від 14.12.2015 р.

ЗАТВЕРДЖЕНО
методичною радою академії зв’язку.

Протокол № від . .2016 р.

Лабораторна робота № 21

Робота з текстовими файлами

Мета роботи: набути практичних навиків програмного створення та редагування текстових файлів засобами Visual C++.

Теоретичні відомості

Загальні відомості про файли

Файлами є іменовані області пам'яті на зовнішньому носії, призначені для довготривалого зберігання інформації. Файли мають ***імена*** й є організовані в ієрархічну деревоподібну структуру з ***каталогів*** (тек) і простих файлів.

Часто використовується така метафора: якщо уявляти собі файли як книжки (лише зчитування) і блокноти (зчитування й записування), які стоять на полиці, то відкриття файла – це вибір книжки чи блокнота за заголовком на його обкладинці й відкриття обкладинки (на першій сторінці). Після відкриття можна читати, дописувати, викреслювати і правити записи, перегортати книжку. Сторінки можна зіставити з блоками файла, а полицю з книжками – з каталогом.

Для доступу до даних файла з програми в ній слід прописати функцію відкриття чи створення цього файла, тим самим встановити зв'язок між ім'ям файла і певною файловою змінною у програмі. Доступ до даних файла відбувається з так званої позиції зчитування-записування, яка автоматично просувається при операціях зчитування-записування, тобто файл є видимим послідовно. Існують, щоправда, функції для довільного змінювання цієї позиції.

C++ надає засоби опрацювання двох типів файлів: *текстових* і *бінарних*.

Текстові файли призначено для зберігання текстів, тобто сукупності символьних рядків змінної довжини. Кожен рядок завершується керувальною послідовністю `'\n'`, а розділювачами слів та чисел у рядку є пробіли й символи табуляції. Оскільки вся інформація текстового файла є символьною, програмне опрацювання такого файла полягає в читанні рядків, виокремлюванні з рядка слів і, за потреби, перетворюванні цифрових символьних послідовностей на числа відповідними функціями перетворювання. Створювати і редагувати текстові файли можна не лише в програмі, а і в якому завгодно текстовому редакторі, наприклад, Word, WordPad чи Блокноті.

Бінарні (двійкові) файли зберігають дані у тому самому форматі, в якому вони були оголошені, а їхній вигляд є такий самий, як і в пам'яті комп'ютера. І тому відпадає потреба у використанні розділювачів: пробілів, керувальних послідовностей, а отже, розмір використовуваної пам'яті порівняно з текстовими файлами з аналогічною інформацією є значно меншим. Окрім того, немає потреби у застосуванні функцій перетворювання числових даних. Але кожне опрацювання даних бінарних файлів можливе лише за наявності програми, якій має бути відомо, що саме і в якій послідовності зберігається у цьому файлі.

У цій лабораторній роботі будуть розглянуті основні засоби створення й опрацювання текстових файлів з використанням спеціального простору імен System::IO платформи .NET Framework, хоча слід зазначити, що існують й інші підходи при роботі з файлами.

Деякі класи простору System::IO для роботи з файлами

Простір імен System::IO платформи .NET Framework містить декілька класів, які дозволяють здійснювати операції з потоками даних, файлами і папками (каталогами), такі як читання і записування даних, пошук (визначення і змінення поточної позиції всередині потоку), архівація файлів і каталогів тощо. Приміром, клас `StreamWriter` з цього простору імен має сучасні, потужні і доволі зручні засоби записування даних у файли як потоки даних.

Охарактеризуємо деякі з цих класів.

Клас	Опис
`File`	надає методи для створення, копіювання, видалення, переміщення і відкриття файлів
`FileInfo`	надає методи екземпляра для створення, копіювання, видалення, переміщення і відкриття файлів
`Directory`	надає статистичні методи для створення і переміщення у каталогах
`DirectoryInfo`	надає методи екземпляра для створення і переміщення у каталогах
`Path`	виконує операції щодо відомостей про шлях до файла чи каталогу (літеральний рядок (тип `String`), який вказує розташування файла у файловій системі – адресу каталогу)
`StreamReader`	надає засоби зчитування послідовності символів (байтів) з потоку в певному кодуванні
`StreamWriter`	надає засоби записування послідовності символів (байтів) у потік в певному кодуванні
`BinaryReader`	надає засоби зчитування простих типів даних (двійкових значень)
`BinaryWriter`	надає засоби записування простих типів даних (двійкових значень)

Розглянемо найпоширеніші методи цих класів, які дозволять створити текстовий файл, записати у нього дані, прочитати їх та ін.

Поширені дії при роботі з файлами

Дії	Методи
Створити текстовий файл	`File::CreateText` або `File::Create` `FileInfo::CreateText` або `FileInfo::Create`
Відкрити файл у заданому режимі і доступі	`File::Open` `File::OpenRead` – відкрити файл у режимі читання; `File::OpenWrite` – відкрити файл для записування
Записати у текстовий файл	`Write` – записати у потік текстове подання аргументів; `WriteLine` – записати рядок і символ `'\n'` як ознаку кінця рядка; `File::WriteAllText` – відкрити текстовий файл, записати всі дані як один рядок у файл і закрити файл

Дописати дані у текстовий файл	`File::AppendText`
	`FileInfo::AppendText`
	методи відкривають існуючий файл і дописують в нього текст у кодуванні UTF-8 (якщо файла не існувало, він буде створений)

Дії	Методи
Зчитати з текстового файла	`Read` – зчитати з потоку текстове подання аргументів, записаних у дужках; `ReadLine` – зчитати рядок і символ `'\n'` як ознаку кінця рядка; `File::ReadAllText` – відкрити текстовий файл, зчитати всі рядки файла в один зазначений рядок і закрити файл
Закрити файл (потік)	`Close`
Перемістити позицію файла на `offset` байтів відносно позиції `SeekOrigin`	`Stream::Seek(offset, SeekOrigin)`
Повернути наступний доступний символ, але не використовувати його	`StreamReader::Peek` з цим методом можна організувати цикл для почергового опрацювання вмісту файла: `while (fr->Peek() > -1) . . .`
Перейменувати чи перемістити файл	`File::Move` або `FileInfo::MoveTo`
Видалити файл	`File::Delete` або `FileInfo::Delete`
Копіювати файл	`File::Copy` або `FileInfo::CopyTo`
Визначити наявність файла	`File::Exists`
Отримати дані про розмір файла	властивість `FileInfo::Length`
Отримати розширення імені файла	`Path::GetExtension`
Отримати повний шлях до файла	`Path::GetFullPath`
Отримати ім'я й розширення файла	`Path::GetFileName`
Змінити розширення файла	`Path::ChangeExtension`

Тобто для однієї дії при роботі з файлами існують декілька методів її реалізації. Розглянемо на прикладах деякі з цих методів.

Створити текстовий файл з ім'ям `MyTest.rtf` у поточному каталозі можна командою:

```
StreamWriter^ f = File::CreateText("MyTest.rtf");
```

або командою:

```
TextWriter^ f = gcnew StreamWriter(File::OpenWrite("MyTest.rtf"));
```

Якщо створюваний файл слід розмістити не в поточному каталозі, а в якомусь іншому, то слід зазначити повний шлях до файла:

```
StreamWriter^ f = File::CreateText("c:\\temp\\MyTest.rtf");
```

Записати у файл рядок `s` можна за допомогою методу `WriteLine`:

```
String^ s = "Текст, який буде записано у файл окремим абзацом.";
f->WriteLine(s);
```

Записати відразу цілий набір рядків, приміром вміст елемента `richTextBox1`, у файл з ім'ям `MyTest.rtf` у поточному каталозі, можна командою:

```
File::WriteAllText("MyTest.rtf", richTextBox1->Text);
```

Дописати дані у кінець файла можна командою

```
File::AppendAllText("MyTest.rtf", "Текст, який буде дописано у файл\n");
```

При цьому, якщо файл з таким ім'ям не існував, він буде створений, тобто метод `AppendAllText` і створює файл, і відкриває його, і дописує в нього дані, поєднуючи можливості відразу декількох методів.

Для **зчитування рядків** текстового файла слід спочатку створити екземпляр класу StreamReader, після чого по черзі зчитувати всі рядки з файла:

```
StreamReader^ f = File::OpenText("MyTest.rtf");
String^ s = "";
while ( s = f->ReadLine() ) Console::WriteLine( s );
```

У наведеному циклі умовою припинення зчитування рядків з файла є нульове значення довжини зчитаного рядка. Така конструкція операторів дозволить переглянути вміст всього текстового файла у консольному режимі.

Іншим, більш компактним способом переглядання вмісту відразу всього текстового файла є використання методу File::ReadAllText, який можна використовувати як у консолі, так і для елементів на формі. До того ж, для цього методу не треба попередньо використовувати команду відкриття файла. Так для консолі програмний код для переглядання усього файла може складатися лише з однієї команди:

```
Console::WriteLine(File::ReadAllText(fname));
```

А для проекту з формою, на якій передбачено елемент richTextBox1 для виведення вмісту файла команда може бути такою:

```
richTextBox1->Text = File::ReadAllText(fname);
```

Переміщуватися по файлу можна не лише при зчитуванні даних, а й за допомогою методу Seek , наприклад:

```
f->BaseStream->Seek(0, SeekOrigin::Begin);  // перейти на початок файла
f->BaseStream->Seek(0, SeekOrigin::End);    // перейти у кінець файла
```

Закрити файл доречно командою

```
f->Close();
```

Приклади програм

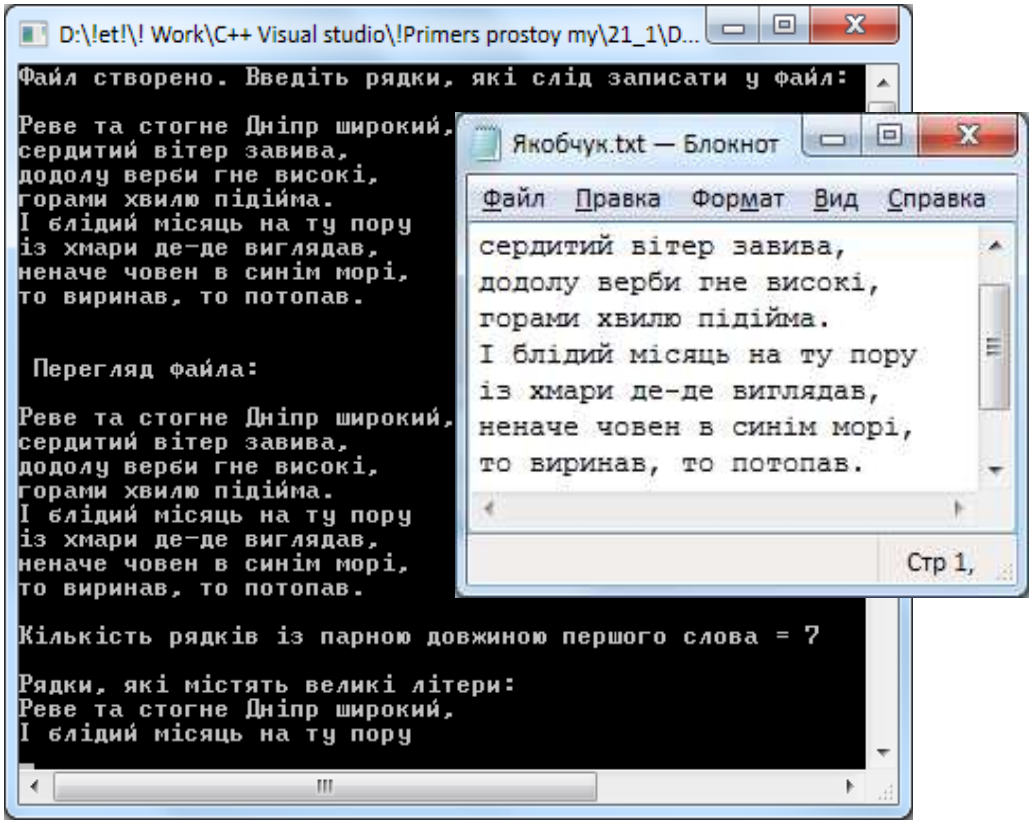
Приклад 1. Створити файл з ім'ям Вашого прізвища і розширенням txt. Ввести рядки з будь-якими словами (бажано не абракадабру) до цього файла (кінець введення – порожній рядок). Переглянути створений файл. Обчислити кількість рядків, в яких перше слово має парну кількість символів (довжину). Вивести рядки, які містять великі літери.

Текст програми:

```
#include "stdafx.h"
using namespace System;
using namespace System::IO;

int main(array<System::String ^> ^args)
{  String ^s, ^fname = "Якобчук.txt";
   StreamWriter^ fw = File::CreateText( fname );   // створити файл
   Console::WriteLine("Файл створено. Введіть рядки, які слід записати у файл:\n");
   do { s=Console::ReadLine();                     // зчитати введений рядок
        fw->WriteLine(s);                          // записати рядок у файл
      }
   while (s!="");
```

```
    fw->Close();                                    // закрити файл
    // відкрити файл для читання рядків і виведення їх на екран
    StreamReader^ fr = File::OpenText( fname );
    Console::WriteLine("\n Перегляд файла:\n");
    while ( s = fr->ReadLine() )                    // читати по черзі рядки з файла
       Console::WriteLine( s );                     // і виводити їх на екран
    fr->BaseStream->Seek(0, SeekOrigin::Begin);     // перейти на початок файла
    int kol=0, ind, n;
    while ( s = fr->ReadLine() )
    { ind = s->IndexOf(" ");                        // шукати індекс пробілу
      if (ind == -1 && s->Length>0) n=s->Length;    // якщо рядок з одного слова
      else n=ind;                                   // n – довжина першого слова
      if (n % 2 == 0) kol++;
    }
    Console::WriteLine("Кількість рядків із парною довжиною першого слова = "+kol);
    fr->BaseStream->Seek(0, SeekOrigin::Begin);     // перейти на початок файла
    Console::WriteLine("\nРядки, які містять великі літери:");
    while ( s = fr->ReadLine() )
    { for (int i=0; i<s->Length; i++)
        if (Char::IsUpper(s[i])) { Console::WriteLine( s ); break; }
    }
    fr->Close();
    Console::Read();
    return 0;
}
```

Результат виконання програми:



Приклад 2. Створити файл з ім'ям Вашого прізвища і розширенням rtf. Ввести рядки з будь-якими словами (бажано не абракадабру) до цього файла (кінець введення – порожній рядок). Переглянути створений файл. Передбачити можливість дописування даних до існуючого файла. Вивести слова текстового файла, які містять більше однієї літери 'а', та обчислити їхню кількість. Вивести рядок, який містить найдовше слово.

Розв'язок. На відміну від попереднього прикладу програми, цю програму організуємо більш структуровано, організувавши різні етапи розв'язання в окремих функціях. Крім того, ця програма дозволить не створювати текстовий файл щоразу заново, а лише за потреби дописувати у нього дані.

Текст програми:

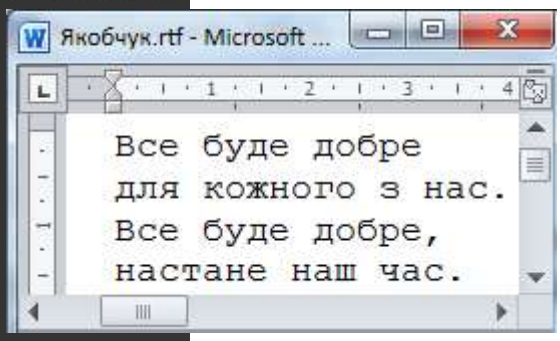
```
#include "stdafx.h"
using namespace System;
using namespace System::IO;

void Записати(String^ fname)
{ String ^s;
  StreamWriter ^fw = File::AppendText( fname ); // відкрити чи створити файл
  try
  { do
    {  s = Console::ReadLine();                 // зчитати введений рядок
       if(s != L"") fw->WriteLine(s);           // записати рядок у файл
    } while (s != "");
  }
  finally
  {  fw->Close(); }                             // закрити файл
  return;
}
void Перегляд(String ^fname)
{ StreamReader ^fr = File::OpenText( fname );   // відкрити файл
  try
  { String ^s = "";
     // читати по черзі усі рядки з файла і виводити їх на екран
    while ( s = fr->ReadLine() ) Console::WriteLine( s );
  }
  finally
  {  fr->Close();  }
  return;
}
void Завдання1(String ^fname)
{ StreamReader ^fr = File::OpenText( fname );
  try
  { String ^s, ^delimStr = " ,.:\t";
    array<Char> ^delimiter = delimStr->ToCharArray( );
    array<String^> ^words;
    int kol=0, ka;
```

```
    Console::WriteLine("\nСлова, які містять більше однієї літери 'a': ");
    while ( s = fr->ReadLine() )
    { s = s->Trim();
      words = s->Split(delimiter, StringSplitOptions::RemoveEmptyEntries);
      for (int i=0; i<words->Length; i++)
      {  for (int j=ka=0; j<words[i]->Length; j++)
           if (words[i][j] == L'a' || words[i][j] == L'a' ||
               words[i][j] == L'A' || words[i][j] == L'A')  ka++;
         if(ka > 1) { kol++; Console::WriteLine(words[i]);}
    } }
    Console::WriteLine("Кількість слів, які містять більше однієї літери 'a' = "+kol);
  }
  finally
  { fr->Close();  }
  return;
}
void Завдання2(String^ fname)
{
  StreamReader ^fr = File::OpenText( fname );    // відкрити файл
  try
  { int max=0, n, k=0;
    String ^s,^slovo, ^delimStr = " ,.:\t";
    array<Char> ^delimiter = delimStr->ToCharArray( );
    array<String^> ^words;
    while ( s = fr->ReadLine() )
    { k++;                              // номер поточного рядка у файлі
      s = s->Trim();                    // видалити початкові і кінцеві пробіли
      words = s->Split(delimiter, StringSplitOptions::RemoveEmptyEntries);
      for (int i=0; i<words->Length; i++)
       if (words[i]->Length > max)
       { max = words[i]->Length;        // запам'ятати довжину найдовшого слова
         slovo = words[i];              // запам'ятати найдовше слово
         n = k;                         // і номер рядка з найдовшим словом
       }
    }
    fr->BaseStream->Seek(0, SeekOrigin::Begin);  // перейти на початок файла
    k=0;                                         // номер поточного рядка
    Console::WriteLine("\nРядок з найдовшим словом - "+slovo+
                       " (довжина - "+slovo->Length+"):");
    while ( s = fr->ReadLine() )
    { k++;
      if (k == n) { Console::WriteLine( s ); break; }
  } }
  finally
  { fr->Close();  }
  return;
}
```

```
int main(array<System::String^> ^args)
{
   String ^fname = "Якобчук.rtf";
   if ( !File::Exists( fname ))
   Console::WriteLine("Файл створено. Введіть рядки, які слід записати у нього:\n");
   else Console::WriteLine("Файл існує. Введіть рядки, які слід дописати у нього.
                \nЯкщо не треба дописувати нові рядки, двічі натисніть Enter.\n");
   Записати(fname);
   Console::WriteLine("Перегляд файла:\n");
   Перегляд(fname);
   Завдання1(fname);
   Завдання2(fname);
   Console::Read();
   return 0;
}
```

Результат виконання програми:

```
Файл створено. Введіть рядки, які слід записати у нього:

Все буде добре
для кожного з нас.
Все буде добре,
настане наш час.

Перегляд файла:

Все буде добре
для кожного з нас.
Все буде добре,
настане наш час.

Слова, які містять більше однієї літери 'а':
настане
Кількість слів, які містять більше однієї літери 'а' = 1

Рядок з найдовшим словом - кожного (довжина - 7):
для кожного з нас.
```




```
Файл існує. Введіть рядки, які слід дописати у нього.
Якщо не треба дописувати нові рядки, двічі натисніть Enter.


Перегляд файла:

Все буде добре
для кожного з нас.
Все буде добре,
настане наш час.
Я пам'ятаю час, коли лиш починався світ
Хто міг, той підіймався та йшов.
Ішов собі високо в гори, взявши у похід
Свою надію сильну, як любов.

Слова, які містять більше однієї літери 'а':
настане
пам'ятаю
Кількість слів, які містять більше однієї літери 'а' = 2

Рядок з найдовшим словом - підіймався (довжина - 10):
Хто міг, той підіймався та йшов.
```

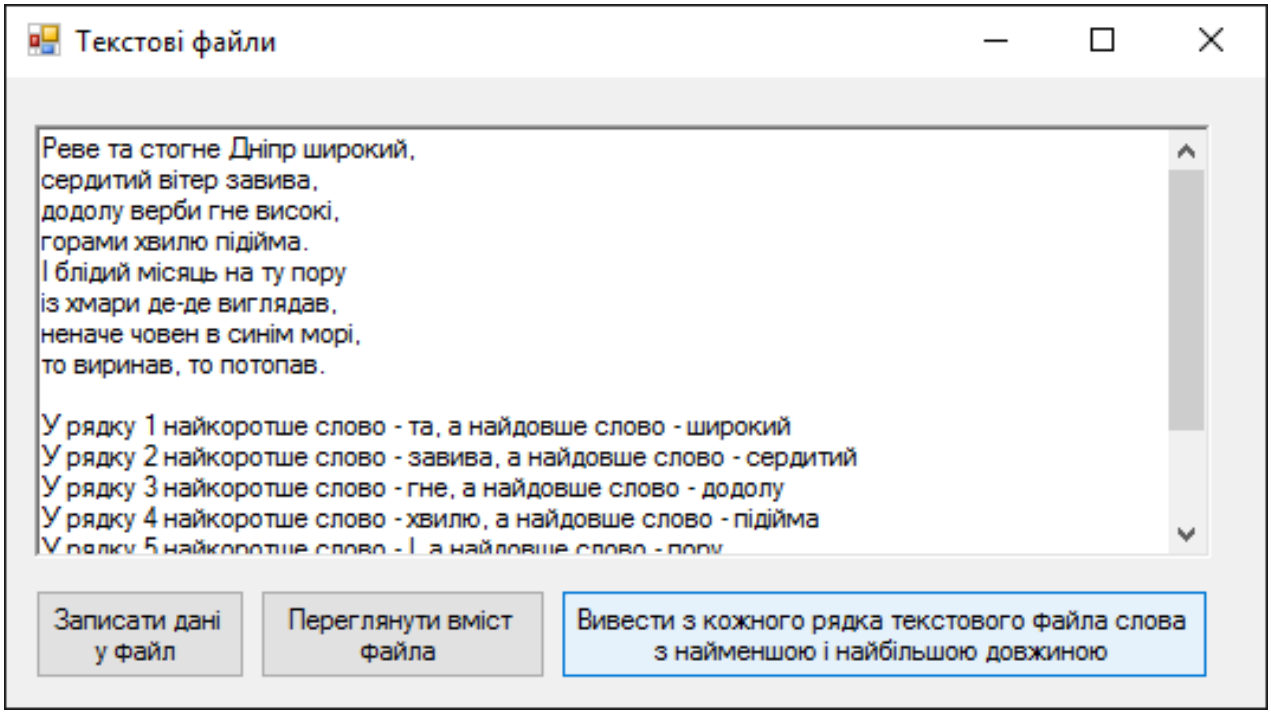
Приклад 3. Створити текстовий файл, у який ввести рядки з будь-якими словами (бажано не абракадабру). Передбачити можливість переглядання створеного файла та вивести з кожного рядка найкоротше та найдовше слова.

Розв'язок. На відміну від попередніх прикладів програм, цю програму створимо з формою.

Текст програми:

```
. . . . . .
using namespace System::IO;
. . . . . .
// Записати дані у файл
private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e)
{
  File::WriteAllText("example.txt", richTextBox1->Text,
                                          System::Text::Encoding::UTF8);
}
// Переглянути вміст файла
private: System::Void button2_Click(System::Object^ sender, System::EventArgs^ e)
{
  richTextBox1->Text = (File::ReadAllText("example.txt"));
}
// Вивести з кожного рядка файла слова з найменшою і найбільшою довжиною
private: System::Void button3_Click(System::Object^ sender, System::EventArgs^ e)
{
 StreamReader ^f = File::OpenText( "example.txt" );
 array<Char> ^separator = { ' ','.',',','!','?','\t' };
 try
 { String ^s,^smin, ^smax; int kmin,kmax,k=0;
   while (s = f->ReadLine())
   {
     if (s->Length>0)
     { k++;
       array<String^> ^subs = s->Split(separator,
       StringSplitOptions::RemoveEmptyEntries);
       kmin=kmax=subs[0]->Length; smin=smax=subs[0];
       for(int i=0;i<subs->Length;i++)
       {  if (kmin > subs[i]->Length) smin=subs[i];
          if (kmax < subs[i]->Length) smax=subs[i];
       }
      richTextBox1->Text+="У рядку "+k.ToString()+" найкоротше слово - " +
                          smin+", а найдовше слово - "+smax+"\n";
   } }
 }
 finally
 { f->Close();
 }
}
```

Результат виконання програми:



Питання для самоконтролю

1) Який файл називають текстовим?
2) У чому полягає відмінність між бінарними і текстовими файлами?
3) Назвати методи файлового записування-зчитування даних.
4) Чим відрізняються методи `Write` і `WriteLine`?
5) Записати команду для створення текстового файла.
6) Записати команду для видалення текстового файла.

Лабораторне завдання

1) У протоколі лабораторної роботи надати відповіді на контрольні питання.

2) У протоколі лабораторної роботи написати мовою С++ програмний код для створення текстового файла з ім’ям Вашого прізвища і розширенням `rtf`, заповнення файла даними (бажано не абракадаброю), переглядання вмісту сформованого файла та розв’язання індивідуальних завдань, які вибрати з табл. 21.1 … 21.2.

3) Створити на комп’ютері програмні проекти у середовищі Visual C++ для реалізації написаних програм.

Таблиця 21.1

Індивідуальні завдання середнього рівня складності

№ вар.	Завдання
1	Обчислити кількість рядків текстового файла, довжина яких більша за 20
2	Вивести на екран рядки текстового файла, які не містять круглі дужки
3	Обчислити кількість рядків, які починаються і закінчуються на одну й ту саму літеру
4	Вивести на екран рядки текстового файла, які не містять розділові знаки
5	Обчислити кількість рядків текстового файла, які містять дефіс
6	Обчислити кількість розділових знаків у текстовому файлі
7	Вивести на екран рядки текстового файла, які не містять великих літер
8	Обчислити кількість цифр у текстовому файлі
9	Вивести на екран рядки текстового файла, які мають парну довжину
10	Обчислити кількість рядків текстового файла, які не містять цифр
11	Вивести на екран рядки текстового файла, в яких є слово "C++"
12	Обчислити середню довжину рядків текстового файла
13	Вивести на екран найдовший рядок текстового файла
14	Обчислити кількість рядків, які містять лише одне слово
15	Вивести на екран найкоротший рядок текстового файла
16	Обчислити кількість рядків файла, які починаються з великої літери
17	Вивести на екран рядки текстового файла, які містять знак оклику
18	Обчислити кількість рядків, які закінчуються не на розділовий знак
19	Вивести на екран рядки текстового файла, які не містять крапок
20	Вивести на екран всі рядки текстового файла, видаливши з них зайві пробіли: початкові, кінцеві і подвійні
21	Вивести на екран рядки текстового файла, які містять цифри
22	Обчислити кількість рядків у текстовому файлі, які закінчуються крапками
23	Вивести на екран всі рядки, переставивши символи у зворотному напряму
24	Вивести на екран всі рядки, перетворивши їхні літери до верхнього регістра
25	Вивести на екран рядки текстового файла, які не містять літер ‘a’
26	Обчислити найбільшу кількість пробілів у рядках текстового файла
27	Вивести на екран номера рядків, які містять літеру ‘R’
28	Обчислити кількість рядків у текстовому файлі з непарною довжиною
29	Вивести на екран рядки текстового файла, які містять рівно 3 слова
30	Обчислити кількість рядків текстового файла, які починаються з літери ‘A’

Таблиця 21.2

Індивідуальні завдання підвищеного рівня складності

№ вар.	Завдання
1	Вивести всі слова, які починаються з літери ‘a’ чи ‘A’, обчислити їх кількість
2	Вивести всі слова файла, довжина яких понад 5, та обчислити їх кількість
3	Вивести слова, які розпочинаються й закінчуються на одну й ту саму літеру
4	Вивести найдовше слово у текстовому файлі та його довжину
5	Вивести з кожного рядка текстового файла слово з найменшою довжиною
6	Вивести слова файла, довжина яких менше 6 символів, та їх кількість
7	Вивести слова текстового файла, які містять літеру ‘z’, та їх кількість
8	Вивести слова, які розпочинаються з великої літери, та визначити їх кількість
9	Вивести слова текстового файла, після яких є кома, та визначити їх кількість
10	Вивести слова файла, які розпочинаються з малої літери, та їх кількість
11	Вивести слова текстового файла з парною довжиною та їх кількість
12	Вивести слова текстового файла, які містять цифри, та їх кількість
13	Вивести з кожного рядка текстового файла слово з найбільшою довжиною
14	Вивести найкоротше слово у текстовому файлі та його довжину
15	Вивести рядок текстового файла, який містить найдовше слово
16	Вивести слова текстового файла, які не містять літеру ‘o’, та їх кількість
17	Вивести слова файла, які містять лише великі літери, та їх кількість
18	Вивести слова файла, які містять лише латинські літери, та їх кількість
19	Обчислити середню довжину слів у текстовому файлі
20	Вивести рядок текстового файла, який містить найкоротше слово
21	Вивести слова файла з довжиною більше семи символів та їх кількість
22	Вивести слова файла, які містять щонайменше дві літери ‘a’, та їх кількість
23	Обчислити середню довжину слів у перших трьох рядках текстового файла
24	Вивести слова текстового файла, які починаються на голосну літеру, та обчислити їх кількість
25	Вивести слова текстового файла з непарною довжиною та їх кількість
26	Вивести слова текстового файла, довжина яких кратна 3, та їх кількість
27	Вивести рядки текстового файла, які містять слова лише з великих літер
28	Вивести рядки текстового файла, які містять слова лише з латинських літер
29	Обчислити кількість слів текстового файла, які містять лише літери кирилиці
30	Вивести слова текстового файла, які закінчуються на голосну літеру, та обчислити їх кількість

Лабораторна робота № 22

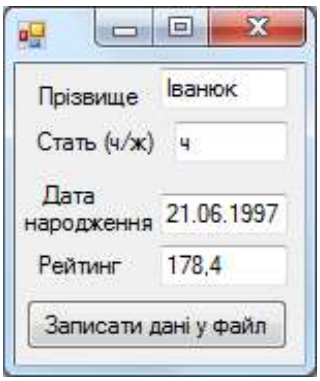
Опрацювання даних типу "дата-час" у текстових файлах

Мета роботи: набути практичних навиків програмного опрацювання даних типу "дата-час".

Теоретичні відомості

Специфіка зберігання різнотипних даних у текстових файлах

C++ надає можливість зберігати дані у текстових файлах не лише у вигляді слів і речень, а й записувати у них числові значення, дані типу "дата-час" тощо. Наприклад, у файл можна записати відомості про студентів: прізвище (рядок – тип `String` або `char*`), стать (символ – тип `Char` або `char`), дата народження (тип `DateTime`), середній бал сесії (дійсне число). Дані про кожного студента зручно зберігати в окремих рядках файла. Так, записування даних про одного студента, які вносяться з чотирьох компонентів `textBox`, слід поєднати в один рядок і дописати цей рядок у файл до даних про інших студентів:




```
File::AppendAllText(fname,textBox1->Text+"\t "+
      textBox2->Text+"\t " +textBox3->Text+"\t "+textBox4->Text+"\n");
```

Якщо в подальшому доведеться зчитувати й аналізувати дані, приміром порівнювати успішність студентів, доведеться розбивати зчитані рядки на частини (слова), виокремлюючи необхідні дані за допомогою функції `Split`. При цьому в якості сепаратора (роздільника) можна зазначити символи пробілу і табуляції. А вже після цього використовувати функції перетворювання даних з рядків до відповідних типів, приміром числових. Слід зазначити, що всередині частин, на які розбивається рядок, не має бути пробілів, тобто якщо поле містить пробіли перед записуванням у файл їх треба замінити, наприклад, на символ підкреслення.

Типи даних "дата-час"

Платформа .NET Framework для роботи з даними "дата-час" має спеціальні типи: `DateTime`, `DateTimeOffset`, `TimeSpan` та ін.

DateTime

`DateTime` – базовий засіб для роботи з датою і часом, який дозволяє визначити конкретні дату і час доби, при чому можна працювати або лише з датою – "27.11.2014", або лише з часом – "11:25:00", або і з датою і з часом одночасно – "27.11.2014 11:25:00". З датами і часом можна виконувати арифметичні дії, наприклад додати до конкретної дати шість місяців. Тип `DateTime` специфічний тим, що значення дати і часу однозначно ідентифікує конкретний момент часу в конкретному часовому поясі і неоднозначно веде себе при зміні часових поясів (в такому разі використовують `DateTimeOffset`).

DateTimeOffset

Значення типу DateTimeOffset завжди однозначно визначає момент часу з урахуванням часового поясу. Значення DateTimeOffset не пов'язане з конкретним часовим поясом, але може бути створене з будь-якого часового поясу. DateTimeOffset доречний для використання, коли треба однозначно ідентифікувати конкретний момент часу, наприклад, при веденні журналу часу транзакцій чи подій системи, особливо коли передбачається перенесення даних з одного ПК на інший.

Тип DateTimeOffset включає більшість функцій типу DateTime, але не замінює цей тип. Використання значень DateTimeOffset більш поширене, аніж DateTime, і розглядається як тип "дата-час" за замовчуванням для розробки програм.

TimeSpan

TimeSpan призначений для роботи з інтервалом часу (тривалістю часу або витраченим часом), який вимірюється як додатне або від'ємне число днів, годин, хвилин і секунд (±д.чч:мм:сс). Максимальною одиницею часу, яку використовує TimeSpan для вимірювання тривалості, є день. Інтервали часу змінюються днями для узгодженості, оскільки кількість днів у більш великих одиницях часу, наприклад місяцях чи то роках, варіюється.

TimeSpan може також використовуватися для подання часу дня, але тільки якщо час не пов'язаний з визначеною датою. Інакше слід використовувати DateTime або DateTimeOffset.

Основні засоби для опрацювання дати і часу

Деякі конструктори

DateTime(Int32, Int32, Int32) – ініціалізує новий екземпляр структури DateTime вказаними значеннями року, місяця й дня, наприклад:

```
DateTime t1=DateTime(2015,11,9);          // 09.11.2015 0:00:00
DateTimeOffset t2=DateTime(2015,11,9);    // 09.11.2015 0:00:00 +02:00
```

DateTime(Int32, Int32, Int32, Int32, Int32, Int32) – ініціює новий екземпляр структури DateTime вказаними значеннями року, місяця, дня, години, хвилини та секунди, наприклад:

```
DateTime d=DateTime(2015,11,9,8,45,00);
Console::WriteLine(d.ToString());          // 09.11.2015 8:45:00
Console::WriteLine(d.ToString("g"));       // 09.11.2015 8:45
Console::WriteLine(d.ToString("t"));       // 8:45 – лише час
Console::WriteLine(d.ToString("d"));       // 09.11.2015 – лише дата
Console::WriteLine(d.ToString("dddd"));    // понеділок
Console::WriteLine(d.ToString(L"MM/dd/yy HH:mm")); // 11.09.15 08:45
Console::WriteLine("{0:dd.MM.yy dddd}",d); // 09.11.2015 понеділок
Console::WriteLine("{0:dddd d MMMM yyyy HH:mm zzz}", d);
// понеділок 9 листопада 2015 8:45 +02:00
```

Заодно було показано різні можливості форматного виведення значення дати і часу за різних потреб.

TimeSpan(Int32, Int32, Int32) – ініціалізує новий екземпляр структури TimeSpan вказаними значеннями годин, хвилин і секунд, наприклад:

```
TimeSpan t=TimeSpan(17,45,00);             // 17:45:00
```

Основні властивості DateTime та DateTimeOffset

Date – з повного значення "дата-час" повертає дату, а значення часу обнулює, наприклад:

```
DateTime d = DateTime::Parse("04.12.2014 17:45:00");
DateTime dateOnly = d.Date;
Console::WriteLine(dateOnly.ToString());      // 04.12.2015 00:00:00
Console::WriteLine(dateOnly.ToString("d")); // 04.12.2015
```

Day – з повного значення "дата-час" повертає ціле значення дня місяця:

```
int day = d.Day;                              // 04.12.2015
Console::WriteLine(day);                      // day = 5
```

Month – зі значення "дата-час" повертає ціле значення місяця від 1 до 12.

Year – зі значення "дата-час" повертає ціле значення року.

Hour – зі значення "дата-час" повертає ціле значення годин від 0 до 23.

Minute – зі значення "дата-час" повертає ціле значення хвилин від 0 до 59.

DayOfWeek – повертає день тижня як одне зі значень діапазону Sunday – Saturday, які в разі переведення до цілих чисел, відповідають діапазону 0 – 6:

```
DateTime d = DateTime::Parse("04.12.2015 17:45:00");
Console::WriteLine(d.DayOfWeek);              // Friday
```

Today – повертає поточну системну дату (без часу).

Now – повертає поточну системну дату і час.

```
DateTime t = DateTime::Today;                 // 09.11.2015 0:00:00
DateTime t = DateTime::Now;                   // 09.11.2015 10:52:36
```

Ticks – повертає ціле число (тип Int64) тактів, які складають зазначені дату і час. До речі, один такт – це 100 наносекунд або одна десятимільйонна секунди. Значення властивості **Ticks** для зазначеного екземпляра означає кількість 100-наносекундних інтервалів, які пройшли з 00:00:00 годин 1 січня 0001 року.

Основні методи

Add – додає до дати і часу (типу DateTime, DateTimeOffset або TimeSpan) значення типу TimeSpan і повертає новий об'єкт DateTime, наприклад:

```
DateTime d1 = DateTime::Now;
Console::WriteLine(" {0:dd.MM.yy dddd}", d1); // 09.11.15 понеділок
TimeSpan duration( 100, 0, 0, 0 );             // 100 днів
DateTime d2 = d1.Add( duration );
Console::WriteLine(" {0:dd.MM.yy  dddd}", d2); // 17.02.16 середа

TimeSpan t = TimeSpan::Parse("15:45");
t = t.Add(t);  // 15:45 + 15:45 = 31.5 година = 1.07:30 (один день і 7.5 годин)
```

AddDays – додає до дати і часу (типу DateTime або DateTimeOffset) зазначену кількість днів (дійсне число) і повертає новий об'єкт DateTime, наприклад:

```
DateTime d = DateTime::Parse("05.12.2015 17:45:00");
d = d.AddDays(2.5);                           // 08.12.2015 5:45
```

AddMonths – додає до дати і часу (типу DateTime або DateTimeOffset) зазначену кількість місяців (дійсне число) і повертає новий об'єкт DateTime.

AddYears – додає до дати і часу (типу DateTime або DateTimeOffset) зазначену кількість років (дійсне число) і повертає новий об'єкт DateTime.

AddHours – додає до дати і часу (типу DateTime або DateTimeOffset) зазначену кількість годин (дійсне число) і повертає новий об'єкт DateTime.

AddMinutes – додає до дати і часу (типу DateTime або DateTimeOffset) зазначену кількість хвилин (дійсне число) і повертає новий об'єкт DateTime.

Compare – порівнює дві дати (типу DateTime, DateTimeOffset або TimeSpan) і повертає ціле число (додатне, якщо перша дата більша, ніж друга, і від'ємне, якщо навпаки), наприклад:

```
int n = DateTime::Compare(d1,d2);
if (n < 0) Console::WriteLine("d1 раніш d2");
else
 if (n > 0) Console::WriteLine("d1 пізшніш d2");
 else Console::WriteLine("d1 = d2");
```

DaysInMonth – повертає кількість днів у зазначеному місяці 28(29), 30 чи 31:

```
int n = DateTime::DaysInMonth(2000,2);        // n = 29
```

Equals – визначає, чи є однаковими дві дати (типу DateTimeOffset, DateTime або TimeSpan). Існують різні форми цього методу з одним і двома аргументами, які є схожими, оскільки повертають логічне значення true чи false. Так, наступні дві команди по суті є ідентичними:

```
bool n = d1.Equals(d2);
bool n = DateTime::Equals(d1,d2);
```

Parse(String) – перетворює рядок на дату типу DateTime:

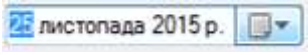
```
DateTime d = DateTime::Parse("25.12.15 17:45");
```

Subtract – віднімає значення аргументу – дати і часу (тип DateTime) або інтервалу часу (тип TimeSpan) і повертає нове значення:

```
DateTime d1, d2, d3, d4, d5; TimeSpan diff1, diff2;
d1 = DateTime( 2015, 6, 20, 22, 15, 0 );     // 20.06.2015 22:15
d2 = DateTime( 2015, 12, 20, 12, 2, 0 );     // 20.12.2015 12:02
d3 = DateTime( 2015, 12, 30, 18, 42, 0 );    // 30.12.2015 18:42
diff1 = d2.Subtract( d3 );                   // diff1=d2-d3 = -10.06:40
d4 = d3.Subtract( diff1 );                   // 10.01.2016 01:22
diff2 = d3 - d2;                             // diff2=d3-d2 = 10.06:40
d5 = d1 - diff2;                             // 10.06.2015 15:35
```

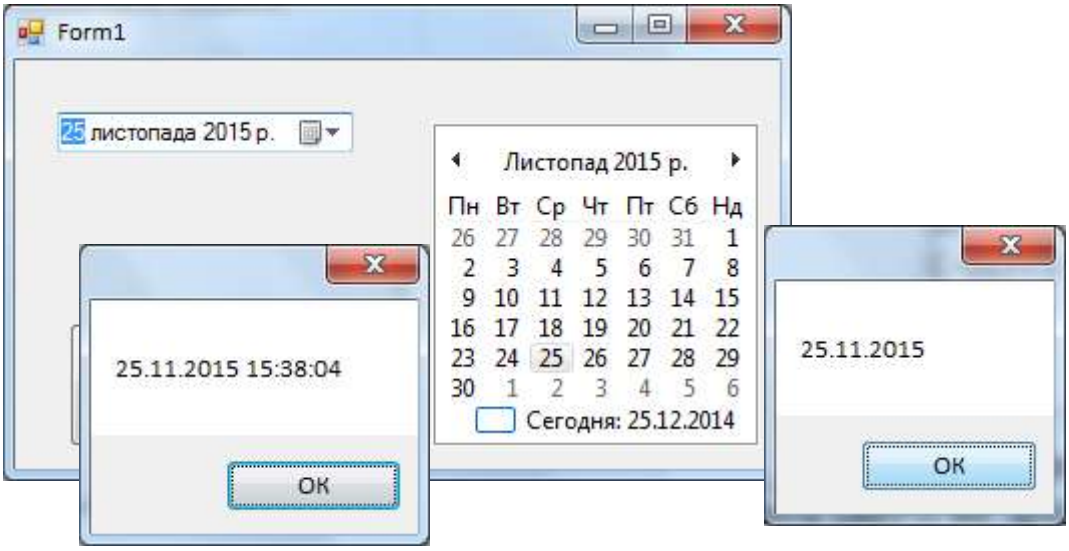
ToString() – перетворює значення дати і часу на рядок. Приклади використання цього методу з можливостями форматного виведення значень дати і часу розглядались на с. 16.

Елементи DateTimePicker та MonthCalendar

Елемент DateTimePicker Windows Forms дозволяє користувачеві вибрати значення дати і часу зі списку у вигляді календарика, при чому сам календарик розкривається при натисканні кнопки зі стрілкою, розташованої праворуч від значення поточної дати. 

Елемент Windows Forms MonthCalendar надає користувачам зрозумілий графічний інтерфейс для перегляду і вибору одного чи навіть декількох значень дати.

```
private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e)
{ DateTime d1 = dateTimePicker1->Value;
  MessageBox::Show(d1.ToString());
  MessageBox::Show(monthCalendar1->SelectionEnd.ToString("d"));
}
```



Приклади програм

Приклад 1. Ввести дату кінцевого терміну придатності деякого продукту та визначити, через скільки днів цей термін скінчиться, порівнявши його з поточною датою.

Текст програми:

```
#include "stdafx.h"
using namespace System;
int main(array<System::String ^> ^args)
{  String^ dstr;
   Console::WriteLine("Ввести термін придатності");
   dstr = Console::ReadLine();
   DateTime dateTerm, dateNow;
   dateTerm = DateTime::Parse(dstr);          // Перетворити рядок на дату
   dateNow = DateTime::Now;                   // Визначити поточну дату
   if (dateTerm > dateNow)                    // Якщо кінцевий термін ще не настав,
   {  TimeSpan days = dateTerm - dateNow;     // обчислити різницю дат
      Console::WriteLine ("Термін придатності скінчиться через " + days.Days
                          + " дня(ів)");
   }  else Console::WriteLine("Продукт прострочений");
   Console::Read();
   return 0;
}
```

Результат виконання програми:

```
Ввести термін придатності
22.12.2014
Термін придатності скінчиться через 123 дня(ів)
```

```
Ввести термін придатності
12.05.2013
Продукт прострочений
```

Приклад 2. Створити текстовий файл з інформацією про дні народження друзів: прізвище, ім'я, дата народження. Переглянути файл. Визначити найстаршого з друзів, а також тих, хто святкуватиме день народження у наступному місяці.

Текст програми:

```
using namespace System;
using namespace System::IO;
void WriteToFile(String^ fname)     // Формування файла з даними
{ String ^surn, ^name, ^birthd;
  do
  { Console::Write("Введіть прізвище ");     surn=Console::ReadLine();
    if(surn != "")
    { Console::Write("Введіть ім'я ");       name=Console::ReadLine();
      Console::Write("Введіть дату народження "); birthd=Console::ReadLine();
      File::AppendAllText(fname, surn+" \t" +name+" \t"+birthd+ "\n");
  } } while (surn != "");
}
void MaxAge(String^ fname)          // Пошук найстаршого з друзів
{ Console::WriteLine("\nНайстарший з друзів");
  String ^maxbirth, ^maxsurn, ^maxname, ^surn, ^name, ^birthd, ^str;
  DateTime maxB;
  array<Char> ^separator = { ' ', '\t' };
  StreamReader ^f = File::OpenText( fname );
  try
  { str = f->ReadLine();            // Зчитати перший рядок з файла
    array<String^> ^substrings = str->Split(separator,
    StringSplitOptions::RemoveEmptyEntries);  // Розбити рядок на слова
    maxsurn = substrings[0];
    maxname = substrings[1];
    maxbirth = substrings[2];
    maxB = DateTime::Parse(maxbirth);
    while (str = f->ReadLine())
    { array<String^> ^substrings = str->Split(separator,
            StringSplitOptions::RemoveEmptyEntries);
     surn = substrings[0];
     name = substrings[1];
     birthd = substrings[2];
     if (DateTime::Parse(birthd)<maxB) //Якщо дата народження менша за максимальну
     { maxbirth = birthd;     maxsurn = surn;
       maxname = name;        maxB = DateTime::Parse(maxbirth);}
     }
    Console::WriteLine(maxsurn+" "+maxname+" "+maxbirth );
   }
   finally
   { f->Close();  }
}
```

```
void NextMonth(String^ fname) // Іменинники наступного місяця
{ Console::WriteLine("\nДрузі, які святкуватимуть день народження у наступному місяці");
 Console::WriteLine("Поточний місяць - "+ (DateTime::Now).ToString(L"MMMM"));
 StreamReader ^f = File::OpenText( fname );
 array<Char> ^separator = { ' ', '\t' };
 try
 { DateTime birthd;  String ^scopy, ^str;
   while (str = f->ReadLine())
   { scopy = str;       // Зробити копію рядка, а сам рядок поділити на слова
    array<String^> ^substrings = str->Split(separator,
                                        StringSplitOptions::RemoveEmptyEntries);
    birthd = DateTime::Parse(substrings[2]); // Третє "слово" рядка – дата народження
    if ((birthd.Month == 1 && DateTime::Now.Month == 12) ||
        (birthd.Month == DateTime::Now.Month+1)) Console::WriteLine(scopy);
} }
  finally  { f->Close();  }
}
int main(array<System::String ^> ^args)
{  String^ fname = "birthd.rtf";
   Console::WriteLine("Введіть дані, які будуть записані у файл.
                       Наприкінці двічі натисніть Enter.");
   WriteToFile (fname);              // Формування файла з даними
   Console::WriteLine("\n Вміст файла:\n");
   Console::WriteLine(File::ReadAllText (fname));
   MaxAge (fname);                   // Пошук найстаршого з друзів
   NextMonth (fname);                // Іменинники наступного місяця
   Console::ReadLine();
   return 0;
}
```

Результат виконання програми:

```
Введіть дані, які будуть дописані у файл. Наприкінці двічі натисніть Enter.
Введіть прізвище Бутур
Введіть ім'я Юрій
Введіть дату народження 05.08.1999
Введіть прізвище

 Вміст файла:

Ткачук  Олег     11.02.1998
Баранов Ігор     25.07.1997
Василюк Ольга    21.01.1996
Петрова Ірина    02.12.1998
Харчук  Іван     31.01.1999
Ященко  Захар    01.01.1997
Коритко Олена    12.03.1997
Званчук Коля     11.09.1998
Гаража  Юлія     04.05.1999
Бутур   Юрій     05.08.1999


Найстарший з друзів
Василюк Ольга 21.01.1996

Друзі, які святкуватимуть день народження у наступному місяці
Поточний місяць - Грудень
Василюк Ольга    21.01.1996
Харчук  Іван     31.01.1999
Ященко  Захар    01.01.1997
```

```
Баранов    Ігор    25.07.1997
Василюк    Ольга   21.01.1996
Петрова    Ірина   02.12.1998
Харчук     Іван    31.01.1999
Ященко     Захар   01.01.1997
Коритко    Олена   12.03.1997
Званчук    Коля    11.09.1998
Гаража     Юлія    04.05.1999
Бутур      Юрій    05.08.1999
```

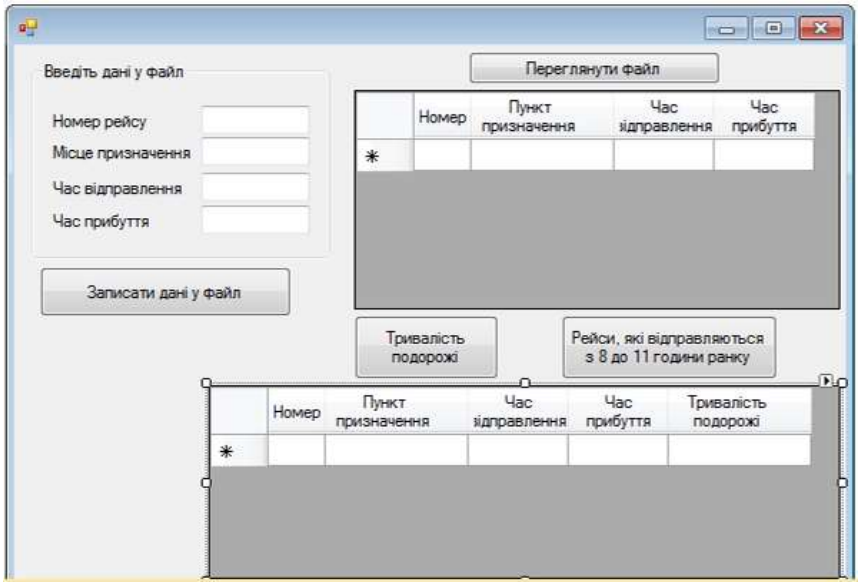
Приклад 3. Створити файл з інформацією про рух автобусів: номер рейсу, місце призначення, час відправлення, час прибуття. Обчислити для кожного з рейсів його тривалість. Знайти автобуси, які відправляються з 8 до 11 години ранку.

Розв'язок. При створенні проекту з формою при її налаштуванні доцільно розмістити на ній:

– для введення даних – чотири textBox з чотирма відповідними label для пояснювальних підписів. З метою угрупування ці вісім елементів можна розташувати на groupBox1, у властивості Text якого записати "Введіть дані у файл";

– чотири кнопки button;

– для переглядання файла – dataGridView1 з чотирма стовпцями, для яких вписати відповідні заголовки стовпців і, за бажанням, задати їх ширину та ін.;



– для почергового виведення результатів завдань про тривалість подорожей і ранкових рейсів – елемент dataGridView2. Щоб не налаштовувати цей елемент "з нуля" доцільно просто скопіювати і вставити елемент dataGridView1. При цьому автоматично створиться dataGridView2 з чотирма вже налаштованими стовпцями і залишиться долучити ще один (п'ятий) стовпець з заголовком "Тривалість подорожі". Оскільки цей п'ятий стовпець не буде потрібним при виведенні даних про ранкові рейси, будемо у програмі тимчасово його вилучати чи долучати.

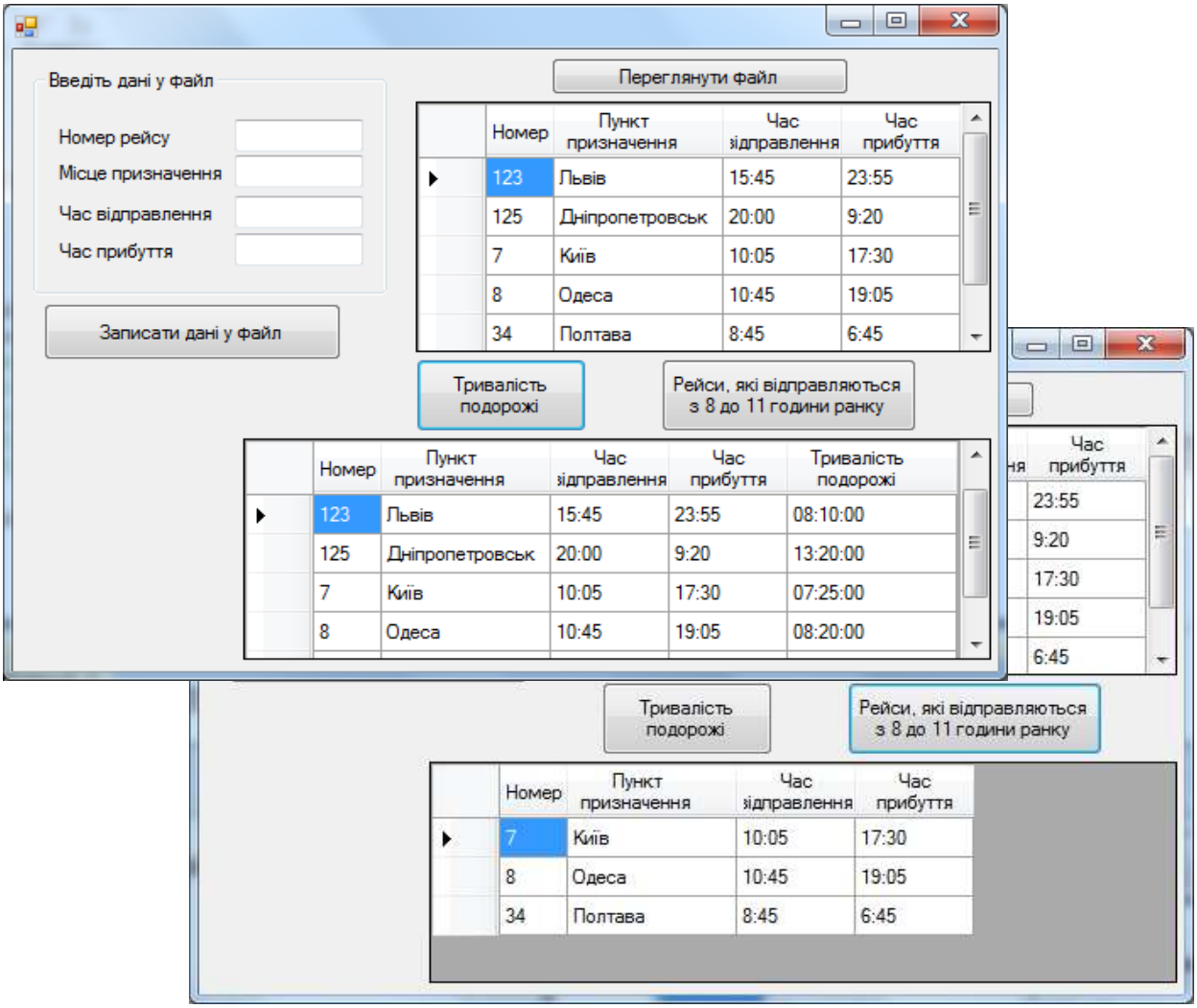
Текст програми:

```
String^ fname; // Глобальна змінна для імені файла
private: System::Void Form1_Load(System::Object^ sender, System::EventArgs^ e)
{
 fname="Buses.rtf"; // Ім'я  файла
 dataGridView1->AllowUserToAddRows = false; //Заборона автоматичного створення рядка
 dataGridView2->AllowUserToAddRows = false;
}
// Кнопка "Записати дані у файл"
private: System::Void button1_Click(System::Object^sender,System::EventArgs^e)
{
   // Записати у файл рядок з даними усіх чотирьох textBox-ів
   File::AppendAllText(fname, textBox1->Text+"\t "+ textBox2->Text+"\t "+
                                    textBox3->Text+"\t "+textBox4->Text+"\n");
   textBox1->Clear(); textBox2->Clear(); textBox3->Clear(); textBox4->Clear();
}
```

```
// Кнопка "Переглянути файл"
private: System::Void button2_Click(System::Object^sender,System::EventArgs^e)
{
  dataGridView1->Rows->Clear();      // Очищення елемента
  if (File::Exists(fname))           // Якщо файл існує, виконати подальші дії
  { String ^str;
    array<Char>^ separator = { ' ', '\t' };   // Роздільники слів у рядку файла
    StreamReader^ f = File::OpenText(fname);  // Відкрити файл для читання
    try
    { // Почергово читати рядки файла, розділяти на слова
      // і виводити їх у клітинки dataGridView1
      while (str = f->ReadLine())
      { array<String^> ^subs = str->Split(separator,
                                    StringSplitOptions::RemoveEmptyEntries);
        dataGridView1->Rows->Add(subs[0],subs[1],subs[2],subs[3]);
      }
    }
    finally
    { f->Close(); }
  }
}
// Кнопка "Тривалість подорожі"
private: System::Void button3_Click(System::Object^ sender,System::EventArgs^ e)
{ dataGridView2->Rows->Clear();
  if(!dataGridView2->Columns->Contains("Column5")) // Якщо п'ятого стовпця немає,
    dataGridView2->Columns->Add("Column5","Тривалість подорожі"); //додати його
  if (File::Exists(fname))           // Якщо файл існує, виконати подальші дії
  { String^ str;
    DateTime departTime, arrivTime;    TimeSpan vputi, t;
    array<Char>^ separator = { ' ', '\t' };
    StreamReader^ f = File::OpenText(fname);
    try
    { while (str = f->ReadLine())
      { array<String^> ^subs = str->Split(separator,
                                    StringSplitOptions::RemoveEmptyEntries);
        departTime = Convert::ToDateTime(subs[2]); // Перетворення рядка на дату
        arrivTime = DateTime::Parse(subs[3]);
        t = TimeSpan(24,0,0);      // t дорівнюватиме 24 годинам
       // Якщо час відправлення більше часу прибуття, додати до часу прибуття 24 години
        if (DateTime::Compare(departTime,arrivTime) > 0)
          vputi = arrivTime.Add(t) - departTime;
        else vputi = arrivTime - departTime;
        dataGridView2->Rows->Add(subs[0],subs[1],subs[2],subs[3],vputi);
    }  }
    finally
    { f->Close(); }
  }
}
```

```
// Кнопка "Рейси, які відправляються з 8 до 11 години ранку"
private: System::Void button4_Click(System::Object^ sender,System::EventArgs^ e)
{ dataGridView2->Rows->Clear();
  if (Column5) dataGridView2->Columns->RemoveAt(4);//Якщо 5-ий стовпець є, видалити його
  if (File::Exists(fname))
  { String^ s;    array<Char>^ separator = { ' ', '\t' };
    StreamReader^ f = File::OpenText(fname);
    DateTime departTime;
    try
    { while (s = f->ReadLine())
      { array<String^> ^subs = s->Split(separator,
                                        StringSplitOptions::RemoveEmptyEntries);
        departTime = DateTime::Parse(subs[2]);
        // Якщо відправлення автобусу між 8 та 11 годинами ранку
        if (departTime >= DateTime::Parse("08:00:00") &&
            departTime <= DateTime::Parse("11:00:00"))
          dataGridView2->Rows->Add(subs[0],subs[1],subs[2],subs[3]);
   } }
   finally  { f->Close();  }
} }
```

Результати виконання програми:



Питання для самоконтролю

1) Які методи дозволяють визначити поточну системну дату?
2) Записати команди, які дозволять збільшити (чи зменшити) дату:
   а) на один день; б) на один тиждень; в) на один місяць?
3) Який метод дозволить перетворити значення типу `DateTime` на рядок?
4) Записати приклад перетворення рядка на значення типу `DateTime`?
5) Який метод дозволить порівняти дві дати? Навести приклад.
6) Чим тип `DateTime` відрізняється від типу `TimeSpan`?

Лабораторне завдання

1) У протоколі лабораторної роботи надати відповіді на контрольні питання.

2) У протоколі лабораторної роботи написати програмний код мовою C++ для формування текстового файла даними (вміст файла зазначено у табл. 22.1), переглядання вмісту сформованого файла та розв'язання індивідуальних завдань з відбирання тих записів файла, які задовольняють певній умові (завдання вибрати з табл. 22.1).

3) Створити на комп'ютері програмні проекти у середовищі Visual C++ для реалізації написаних програм.

Таблиця 22.1

Індивідуальні завдання

№ вар.	Вміст текстового файла	Завдання
1	Список для автоінспекції: відомості про викрадені автомобілі: державний номер, марка автомобіля, колір, дата заяви	Вивести відомості про викрадені автомобілі марки "BMW", які було викрадено понад два роки тому
2	Відомості про сплату комунальних послуг: вулиця, номер будинку, прізвище мешканця, дата сплати, борг	Вивести відомості боржників з боргом понад 500 грн., які не сплачували послуги вже 3 місяці
3	Список людей, які стоять у черзі на встановлення телефону: прізвище, дата подання заяви, пільги ("Так" або "Ні")	Вивести список тих, хто має пільги і стоїть у черзі понад два роки
4	Літній розклад руху потягів: номер потяга, пункт відправлення, пункт призначення, час відправлення	Вивести відомості про потяги напрямку Одеса – Київ, які відправляються з 10:00 до 17:00
5	Журнал подій операційної системи: назва запущеної програми, рівень події (*помилка*, *попередження* тощо), дата події, час події	Вивести дані про помилки та визначити скільки днів пройшло від кожної з помилок на поточний момент часу
6	Список ліній інтенсивного зв'язку метеорного каналу: напрямок зв'язку, час початку (hh:mm) й закінчення (hh:mm) інтенсивного зв'язку, місяць	Вивести відомості про напрямок з найбільшим інтервалом зв'язку у травні-місяці

Продовження табл. 22.1

№ вар.	Вміст текстового файла	Завдання
7	Відомості про медикаменти в аптеці: назва ліків, дата кінцевого терміну придатності, ціна	Вивести дані про медикаменти, термін придатності яких закінчується: 1) у поточному році; 2) в межах 30-ти днів
8	Список співробітників підприємства: табельний номер, прізвище, стать (ч/ж), дата народження, посада	Вивести відомості про усіх співробітників-жінок старше 55-ти років та чоловіків старше 60-ти років
9	Відомості про товари: назва ліків, дата виготовлення, дата кінцевого терміну реалізації, ціна	Вивести відомості про товари з терміном реалізації: 1) менше 10-ти днів; 2) в поточному місяці
10	Розклад телепередач: назва, початок у форматі чч:хх, тривалість у форматі чч:хх (бажано вводити передачі у порядку їх показу)	Визначити для кожної передачі час закінчення, а також вивести відомості про вечірні передачі (з 18:00 до 22:00)
11	Розклад руху приміських потягів: номер потяга, кінцевий пункт призначення, час відправлення, тривалість подорожі	Обчислити час прибуття потягів та вивести відомості про маршрути тривалістю більше трьох годин
12	Список співробітників підприємства: табельний номер, прізвище, посада, дата народження, дата прийому на роботу	Вивести відомості про співробітників, які пропрацювали понад десять років на підприємстві
13	Список святкових днів у календарі: назва свята, дата у форматі (дд.мм.гггг)	Вивести дані про зимові свята та визначити найближче свято
14	Розклад руху літаків: номер рейсу, пункт призначення, час відправлення	Вивести дані про початок і закінчення реєстрації (відповідно за 2 години і 40 хвилин до відправлення)
15	Список студентів групи: прізвище, ім’я, дата народження, середній бал успішності	Визначити вік кожного студента. Знайти наймолодшого студента
16	Репертуар оперного театру: назва вистави, жанр, дата, початок	Вивести відомості про вистави жанру балет, які ще не відбулися, а також дитячі вистави (початок до 15 годин)
17	Майстерня телефонної мережі: прізвище абонента, номер телефону, дата поломки, дата усунення	Вивести дані про поломки у минулому місяці, а також визначити термін усунення кожної поломки у днях
18	Список співробітників підприємства: табельний номер, прізвище, стать, дата народження, посада	Вивести відомості про співробітників, які святкують у цьому році ювілей (кількість років кратна 5)
19	Розклад руху потягів: номер потяга, кінцевий пункт призначення, час відправлення, час прибуття	Визначити найдовший та найкоротший маршрути

Закінчення табл. 22.1

№ вар.	Вміст текстового файла	Завдання
20	Замовлення на стаціонарне підключення до Інтернет: прізвище, номер контактного телефону, дата замовлення	Вивести для кожного замовлення дату можливого підключення, якщо термін складає 5 днів. Вивести усі замовлення минулого місяця
21	Статистика відправлень SMS у телефоні: номер телефону адресата, дата і час відправлення	Вивести дані про SMS, відправлені минулого тижня, а також SMS, відправлені вночі (з 23:00 до 4:00)
22	Список товарів у магазині: назва, виробник, дата виготовлення, термін зберігання у днях	Визначити для кожного продукту дату закінчення терміну зберігання та вивести дані про прострочені продукти
23	Звіт про сплату комунальних платежів: особистий рахунок, прізвище, сума сплати, дата сплати	Вивести дані про сплачені послуги за останні три місяці
24	Список співробітників підприємства: табельний номер, прізвище, дата народження, посада, дата прийому на роботу	Вивести дані про найстаршого за віком співробітника та найменшого за стажем на підприємстві співробітника
25	Розклад руху пасажирських потягів на проміжній станції: номер потяга, назва станції призначення, час прибуття, час відправлення	Вивести дані про стоянки понад 15 хвилин та віднайти найдовшу стоянку
26	Відомості про реєстрацію співробітників перед початком робочого дня: прізвище, посада, час прибуття на роботу	Вивести дані про робітників, які спізнилися (робочий день починається о 9:00), за винятком директора та начальників відділів
27	Результати змагань плавців: прізвище учасника, дистанція (у м), час у форматі чч:хх:сс	Визначити переможця на дистанції 100 м
28	Репертуар кінотеатру: назва фільму, назва залу, початок сеансу у форматі чч:хх, тривалість у форматі чч:хх	Вивести дані про фільми, які починаються увечері з 16:00 до 19:00, а також фільми тривалістю понад 2 години
29	Військкомат: прізвище, ім’я, звання, дата призову, дата звільнення у запас	Вивести дані про військовозобов'язаних, які ще не звільнені (дата звільнення не заповнена)
30	Журнал перезавантажень сервера: дата, час, причина, прізвище відповідального	Вивести відомості про перезавантаження системи протягом минулого тижня та ті перезавантаження, які відбулися вночі з 23:00 до 5:00

Лабораторна робота № 23

Робота з бінарними файлами

Мета роботи: набути практичних навиків програмного створення та редагування бінарних файлів засобами Visual C++.

Теоретичні відомості

Загальні відомості про бінарні файли

Бінарні файли зберігають дані у тому самому форматі, в якому вони були оголошені, і їхній вигляд є такий самий, як і в пам'яті комп'ютера. І тому відпадає потреба у використанні розділювачів: пробілів, керувальних послідовностей, а отже, розмір бінарного файла порівняно з текстовим файлом з аналогічними даними буде значно меншим. Окрім того, немає потреби у застосуванні функцій перетворювання числових даних при зчитуванні-записуванні. Слід зазначити, що опрацювання (перегляд, запис та ін.) бінарних файлів можливе лише з програми, яка знає, що саме і в якій послідовності зберігається у цьому файлі.

У цій лабораторній роботі будуть розглянуті основні засоби створення й опрацювання бінарних файлів з використанням спеціального простору імен System::IO, хоча слід зазначити, що існують й інші підходи при роботі з файлами.

Деякі засоби простору System::IO для роботи з бінарними файлами

Для роботи з бінарними файлами у цій роботі будуть використані класи `File`, `BinaryReader` і `BinaryWriter` з простору імен System::IO платформи .NET Framework, які дозволять створити файл та здійснювати операції зчитування/записування двійкових даних з/у файл як потік даних.

Розглянемо найпоширеніші методи цих класів.

Поширені дії при роботі з файлами

Дії	Методи
Відкривання файла у заданому режимі і доступі	`File::Open`
Записування у файл	`Write` – записує у потік двійкове подання аргументу
Зчитування з файла	`Read` – зчитує з потоку значення аргументу в дужках
Відкриття файла	`File::Open`
Закриття файла (потоку)	`Close`

Переміщення позиції файла на `offset` байтів відносно позиції `SeekOrigin:Begin` або `SeekOrigin::End`	**`BinaryWriter::Seek(offset, SeekOrigin)`**
	Наприклад:
	`f->BaseStream->Seek(0, SeekOrigin::Begin);//на початок файла`
	`f->BaseStream->Seek(0, SeekOrigin::End);  //у кінець файла`

Наприклад, **створити бінарний файл** з ім'ям `Test.dat` у поточному каталозі можна командою:

```
BinaryWriter^ f=gcnew BinaryWriter(File::Open("Test.dat",FileMode::OpenOrCreate));
```

або двома командами:

```
FileStream^ data = gcnew FileStream("Test.dat", FileMode::OpenOrCreate);
BinaryWriter^ fb = gcnew BinaryWriter(data);
```

При цьому режим відкриття OpenOrCreate дозволить не обнуляти вже існуючий файл (якщо такий є), а лише відкрити його.

Якщо створюваний файл слід розмістити не в поточному каталозі, а в якомусь іншому, слід зазначити повний шлях до файла.

Приклади програм

Приклад 1. Створити програму, яка дозволить зберігати у бінарному файлі двійкові дані: номер мобільного телефону, дату підключення (активації) номера, вартість однієї хвилини розмов, назву тарифу, назву мобільного оператора. Крім програмного створення файла і заповнення його даними, передбачити програмний перегляд даних зі створеного файла. Відібрати абонентів, які користуються мобільним зв'язком понад 10 років без зміни номера телефону.

Текст програми:

```
#include "stdafx.h"
using namespace System;
using namespace System::IO;

void Zapis(String^ fname)
{
  BinaryWriter^fb=gcnew BinaryWriter(File::Open(fname,FileMode::OpenOrCreate));
  fb->Seek(0, SeekOrigin::End);
  try
  {
    String ^номер, ^тариф, ^оператор;  double плата;  DateTime d;
    do
    { Console::Write("\nВведіть номер телефона ");
      номер = Console::ReadLine();
      if(номер != "")
      { Console::Write("Введіть дату підключення номера - ");
        d = Convert::ToDateTime(Console::ReadLine());
        Console::Write("Введіть вартість 1 хв. розмов - ");
        плата = Convert::ToDouble(Console::ReadLine());
        Console::Write("Введіть назву тарифу - ");
        тариф = Console::ReadLine();
        Console::Write("Введіть оператора - ");
        оператор = Console::ReadLine();
        fb->Write(номер); fb->Write(d.Ticks); fb->Write(тариф);
        fb->Write(плата); fb->Write(оператор);
      }
    } while (номер != "");
  } finally { fb->Close(); }
}
```

```
void Prosmotr(String^ fname)
{
  if(!File::Exists(fname)) { Console::WriteLine("Файл не знайдено!"); return;}
  BinaryReader^ fb = gcnew BinaryReader(File::OpenRead(fname));
  try
  { Console::WriteLine("\nПерегляд файла:");
    while (fb->BaseStream->Position < fb->BaseStream->Length)
    {
       String^ номер = fb->ReadString();
       DateTime d = DateTime( fb->ReadInt64() );
       String^ тариф = fb->ReadString();
       double плата = fb->ReadDouble();
       String^ оператор = fb->ReadString();
       Console::WriteLine(номер +"\t "+ d.ToString("d") +"\t "+ тариф +"\t "+
         Convert::ToString(String::Format("{0:0.00}",плата)) +"\t "+ оператор);
    }
  }
  finally { fb->Close(); }
}

void ThisYear(String^ fname)
{ if(!File::Exists(fname)) { Console::WriteLine("Файл не знайдено!"); return;}
  BinaryReader^ fb = gcnew BinaryReader(File::OpenRead(fname));
  try
  { Console::WriteLine("\nАбоненти, піключені понад 10 років тому:");
    while (fb->BaseStream->Position < fb->BaseStream->Length)
    {  String^ номер = fb->ReadString();
       DateTime d = DateTime( fb->ReadInt64() );
       String^ тариф = fb->ReadString();
       double плата = fb->ReadDouble();
       String^ оператор = fb->ReadString();
       if ( d.Year+10 <= DateTime::Now.Year )
        Console::WriteLine(номер +"\t "+d.ToString("d") +"\t "+ тариф +"\t  "+
        Convert::ToString(String::Format("{0:0.00}",плата)) +"\t "+ оператор);
  } }
  finally { fb->Close(); }
}

int main(array<System::String^> ^args)
{ String^ fname = "telephons.dat";
  Console::WriteLine(L"Введіть дані, які будуть записані у файл.\n
                       Наприкінці двічі натисніть Enter.");
  Zapis(fname);
  Prosmotr(fname);
  ThisYear(fname);
  Console::ReadLine();
  return 0;
}
```

Результат виконання програми:

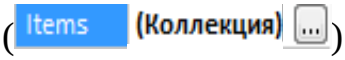
```
Введіть дані, які будуть записані у файл.
Наприкінці двічі натисніть Enter.

Введіть номер телефона 097-977-34-34
Введіть дату підключення номера - 25.04.2008
Введіть вартість 1 хв. розмов - 0,45
Введіть назву тарифу - Вільний день
Введіть оператора - Київстар

Введіть номер телефона

Перегляд файла:
067-800-12-34    21.04.2000      Вільний день    0,45    Київстар
050-266-55-40    30.03.1999      3D ноль         0,50    МТС Україна
063-777-54-32    12.03.2015      Смартфон+       0,35    LifeCell
068-70-444-17    29.09.2012      Для смартфона   0,75    КиївСтар
094-33-79-004    21.09.2011      Non stop        0,55    Інтертелеком
097-977-34-34    25.04.2008      Вільний день    0,45    Київстар

Абоненти, піключені понад 10 років тому:
067-800-12-34    21.04.2000      Вільний день     0,45   Київстар
050-266-55-40    30.03.1999      3D ноль          0,50   МТС Україна
```

Приклад 2. Створити програму, яка дозволить зберігати у бінарному файлі двійкові дані: номер мобільного телефону, назву тарифу, назву мобільного оператора, наявність абонплати, вартість одної хвилини розмов. Крім програмного створення файла і заповнення його даними, передбачити програмний перегляд даних зі створеного файла. Відібрати дані про номери телефонів на Київстарі та обчислити їхню кількість і середню вартість 1 хв. розмов. Відібрати тарифи без абонплати. Визначити найдешевший тариф без абонплати.

Розв'язок. Для налаштування форми на ній слід розмістити відповідні командні кнопки і таблиці (елементи `dataGridView`) для введення-виведення даних файла. Розглянемо докладніше налаштування першого `dataGridView`. Після розміщення його на формі слід почергово створити на ньому п'ять стовпців за кількістю полів у завданні і при їх створенні відразу задати тип даних і заголовки стовпців:

– для 1-го стовпця – у властивості `HeaderText` вписати `Номер телефону`;

– для 2-го стовпця – у властивості `HeaderText` вписати `Назва тарифу`;

– для 3-го стовпця – у властивості `HeaderText` вписати `Оператор`, а для властивості `ColumnType` замість значення за замовчуванням вибрати зі списку значення `DataGridViewComboBoxColumn`. Після цього у властивості `Items` (

) вписати через *Enter* назви операторів, наприклад: МТС, Київстар, Life;

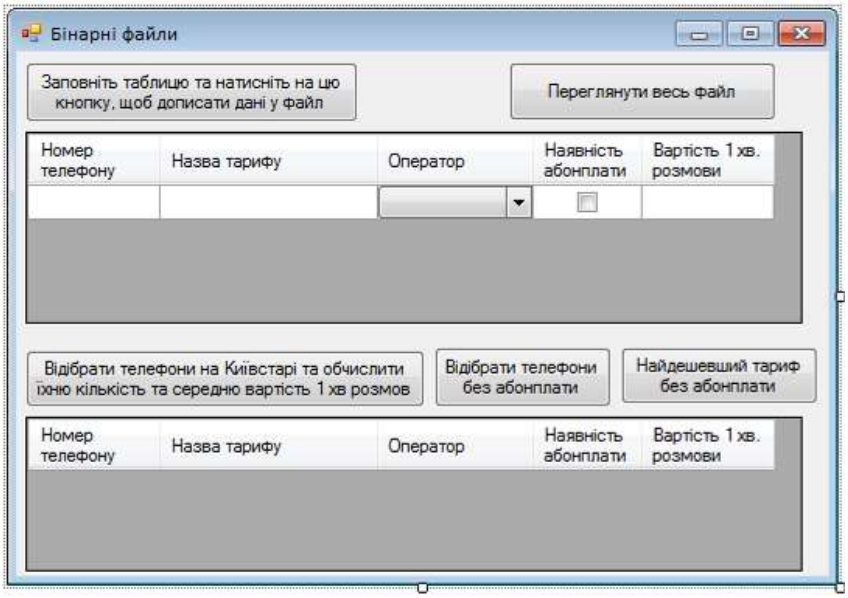
– для 4-го стовпця – у властивості `HeaderText` вписати `Наявність абонплати`, а для властивості `ColumnType` замість значення за замовчуванням вибрати зі списку значення `DataGridViewCheckBoxColumn`;

– для 5-го стовпця – у властивості `HeaderText` вписати `Вартість 1 хв. розмови`.

Можна також для кожного зі стовпців задати ширину у властивості `Width`.

Тепер можна скопіювати і вставити `dataGridView1`, створивши таким чином вже налаштований `dataGridView2`, для якого лише залишилось для властивостей `Visible` та `AllowUserToAddRows` встановити значення `false`.

Після налаштування решти елементів на формі вона набуде вигляду:



Специфікою файла Form1.h з програмним кодом є наявність значної кількості рядків з командами налаштування елементів на формі, які автоматично створюються у цьому файлі при розміщенні нових елементів на формі та встановленні їх властивостей. Тому, щоби вписати команду для зазначення використання простору імен `using namespace System::IO;` з програмними засобами файлового введення-виведення, слід піднятися на початок файла, віднайти рядок `using namespace System;` і під ним вписати зазначений. Після цього спуститися донизу і повернутися до створення програмного коду для п'ятьох кнопок button.

Текст програми:

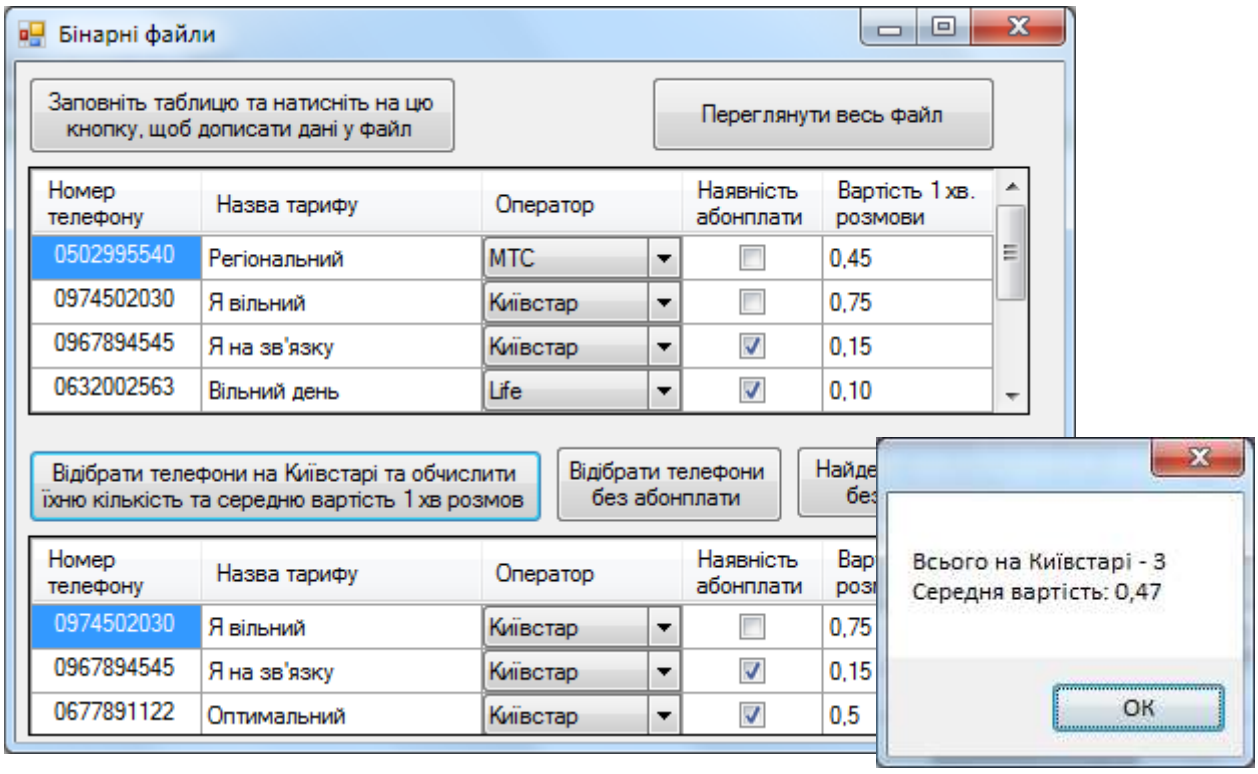
```
. . . . . .
using namespace System;
using namespace System::IO;
. . . . . .
String ^fname; // глобальна змінна  - ім'я файла

private: System::Void Form1_Load(System::Object^ sender, System::EventArgs^ e)
{
  fname = "connect.dat";
}

// Записати дані у файл
private: System::Void button1_Click(System::Object^sender,System::EventArgs^e)
{
  BinaryWriter^ fb=gcnew BinaryWriter(File::Open(fname,FileMode::OpenOrCreate));
  // або замість цієї команди можна скористатися двома такими
  // FileStream^ phons = gcnew FileStream(fname, FileMode::OpenOrCreate);
  // BinaryWriter^ fb = gcnew BinaryWriter(phons);
  fb->Seek(0, SeekOrigin::End); // перейти у кінець файла для дописування даних
  try
  {
    int k=dataGridView1->Rows->GetRowCount(DataGridViewElementStates::Visible);
    for (int i=0; i < k-1; i++)
```

```
    { String ^номер = Convert::ToString(dataGridView1[0,i]->Value);
      // звертатися до клітинки таблиці можна і так: dataGridView1->Rows[i]->Cells[0]->Value
      String ^тариф = Convert::ToString(dataGridView1[1,i]->Value);
      String ^оператор = Convert::ToString(dataGridView1[2,i]->Value);
      bool абонплата = Convert::ToBoolean(dataGridView1[3,i]->Value);
      double vart = Convert::ToDouble(dataGridView1[4,i]->Value);
      fb->Write(номер);       fb->Write(тариф);   fb->Write(оператор);
      fb->Write(абонплата);   fb->Write(vart);
    }
   dataGridView1->Rows->Clear();
  }
  finally
  { fb->Close();  }
}
// Вивести бінарний файл
private: System::Void button2_Click(System::Object^sender,System::EventArgs^e)
 {
   dataGridView1->Rows->Clear();
   if ( File::Exists(fname) == false )
   { MessageBox::Show("file not exists");  return; }
   BinaryReader^ fb = gcnew BinaryReader(File::OpenRead(fname));
   try
   {
     while (fb->BaseStream->Position < fb->BaseStream->Length)
     { String^ номер = fb->ReadString();
       String^ тариф = fb->ReadString();
       String^ оператор = fb->ReadString();
       bool абонплата = fb->ReadBoolean();
       double vart = fb->ReadDouble();
       dataGridView1->Rows->Add(номер,тариф,оператор,абонплата,
                                              String::Format("{0:0.00}",vart));
     }
   }
   finally
   { fb->Close(); }
}
// Відібрати телефони на Київстарі та обчислити їх кількість і середню вартість розмов
private: System::Void button3_Click(System::Object^sender,System::EventArgs^e)
{
  int k=0; double s=0;
  dataGridView2->Visible=true;   // Відобразити таблицю
  dataGridView2->Rows->Clear();  // Очистити таблицю
  if ( File::Exists(fname) == false )
  { MessageBox::Show("file not exists");  return; }
  BinaryReader^ fb = gcnew BinaryReader(File::OpenRead(fname));
  try
  {
    while (fb->BaseStream->Position < fb->BaseStream->Length)
```

```
    { String^ номер = fb->ReadString();
      String^ тариф = fb->ReadString();
      String^ оператор = fb->ReadString();
      bool абонплата = fb->ReadBoolean();
      double vart = fb->ReadDouble();
      if (оператор == "Київстар")
      { dataGridView2->Rows->Add(номер, тариф, оператор, абонплата,
                                          String::Format("{0:0.00}",vart));
        s += vart;
        k++;
   } }
   if (k) s /= k;
   MessageBox::Show("Всього на Київстарі - "+(k).ToString()+
      "\nСередня вартість: "+Convert::ToString(String::Format("{0:0.00}",s)));
 }
 finally
 { fb->Close(); }
}
// Телефони без абонплати
private: System::Void button4_Click(System::Object^sender,System::EventArgs^e)
{
  dataGridView2->Visible=true;  // Відобразити таблицю
  dataGridView2->Rows->Clear(); // Очистити таблицю
  if ( File::Exists(fname) == false )
  { MessageBox::Show("file not exists");  return;}
  BinaryReader^ fb = gcnew BinaryReader(File::OpenRead(fname));
  try
  {
    while (fb->BaseStream->Position < fb->BaseStream->Length)
    { String^ номер = fb->ReadString();
      String^ тариф = fb->ReadString();
      String^ оператор = fb->ReadString();
      bool абонплата = fb->ReadBoolean();
      double vart = fb->ReadDouble();
      if(!абонплата) dataGridView2->Rows->Add(номер, тариф, оператор,
                                  абонплата, String::Format("{0:0.00}",vart));
  } }
  finally
  { fb->Close();  }
}
// Найдешевший тариф без абонплати
private: System::Void button5_Click(System::Object^sender,System::EventArgs^e)
{ dataGridView2->Visible = true;
  dataGridView2->Rows->Clear();
  if ( File::Exists(fname) == false )
  { MessageBox::Show("file not exists");  return; }
  BinaryReader^ fb = gcnew BinaryReader(File::OpenRead(fname));
  try
```

```
  { double vart, min=100; // нереальне число
    String ^номер, ^тариф, ^оператор, ^nommin,^tarmin, ^opmin;
    while (fb->BaseStream->Position < fb->BaseStream->Length)
    { номер = fb->ReadString();
      тариф = fb->ReadString();
      оператор = fb->ReadString();
      bool абонплата = fb->ReadBoolean();
      vart = fb->ReadDouble();
      if(!абонплата && min>vart)
      { min = vart;  nommin = номер;  tarmin = тариф;  opmin = оператор;
    } }
    dataGridView2->Rows->Add(nommin, tarmin, opmin, false, min);
  }
  finally
  { fb->Close(); }
}
```

Результат виконання програми:



Питання для самоконтролю

1) Який файл називають бінарним?
2) У чому полягає відмінність між режимами відкриття файлів `OpenOrCreate` і `Open`?
3) Записати команду для створення бінарного файла.
4) Записати команду для переміщення на початок бінарного файла і ще одну команду для переміщення у кінець цього файла. В яких випадках ці команди будуть доречні для використання?

Лабораторне завдання

1) У протоколі лабораторної роботи надати відповіді на контрольні питання.

2) У протоколі лабораторної роботи написати мовою С++ програмний код для створення бінарного файла з ім'ям Вашого прізвища і розширенням dat, заповнення файла даними, переглядання вмісту сформованого файла та розв'язання індивідуальних завдань, які вибрати з табл. 23.1.

3) Створити на комп'ютері програмний проект у середовищі Visual C++ для реалізації написаної програми.

Таблиця 23.1

Індивідуальні завдання

№ вар.	Вміст бінарного файла	Завдання
1	Список абонентів телефонної станції: прізвище абонента, номер телефону, абонплата за місяць, сума боргу	Вивести відомості про абонентів, які мають борг понад 200 грн.
2	Список принтерів для продажу: тип принтера, фірма-виробник, швидкість роботи (кількість аркушів за хвилину), ціна принтера	Вивести відомості про принтери фірми HP, які друкують понад 10 аркушів за хвилину
3	Список аварійних будинків у місті: вулиця, номер будинку, кількість мешканців у будинку, рік постановки на облік	Вивести відомості про будинки, в яких понад 200 мешканців та які понад 30 років мають аварійний стан
4	Список студентів у групі: номер у журналі, прізвище та ім'я, оцінки з фізики, математики, філософії	Вивести список студентів, які не мають трійок, а також середній бал кожного студента
5	Список робітників підприємства: табельний номер, ПІБ, посада, стать (ч або ж), адреса	Вивести відомості про середній вік чоловіків та жінок, а також скласти список жінок, яким залишився рік до пенсії
6	Дані про автомобілі для продажу: тип автомобіля, тип двигуна, кілометраж пробігу, рік випуску, стартова ціна	Вивести відомості про автомобілі Ford, в яких пробіг становить менше 50000 км і випущених понад два роки тому
7	Список студентів групи: номер у журналі, ПІБ, рейтинг з фізики, математики, інформатики	Вивести студентів, які мають з усіх предметів рейтинг не нижче 95 балів, а також студентів, в яких хоча б з одного предмета є 100 балів
8	Список комп'ютерів, які встановлені в аудиторії: порядковий номер, тип ПК, тактова частота, оперативна пам'ять, ємність диска, рік встановлення у класі	Вивести відомості про ЕОМ, в яких ємність диска менше 80 Гб, а оперативна пам'ять менше 4 Гб

Продовження табл. 23.1

№ вар.	Вміст бінарного файла	Завдання
9	Список товарів у магазині електроніки: код, назва, виробник, країна-виробник, рік випуску, вартість	Вивести дані про холодильники, а також обчислити середню вартість китайських телевізорів
10	Список робітників підприємства: табельний номер, прізвище та ініціали, посада, дата прийому на роботу, оклад	Вивести відомості про програміста, який отримує найбільший оклад, а також дані про всіх менеджерів
11	Список команди "Чорноморець": номер на полі, ПІБ, амплуа (воротар, захисник тощо), кількість забитих і пропущених голів за сезон	Вивести відомості про нападників команди, які не забили жодного гола, та найти воротаря, який пропустив найбільше голів
12	Список книг у книгосховищі: інвентарний номер, назва книги, автор, рік видання, ціна	Вивести книги, назви яких починаються зі слова "Програмування", а також список книг, які зберігаються понад 10 років
13	Список мешканців: вулиця, номер будинку, номер квартири, ПІБ власника, рік останнього капітального ремонту	Вивести всі дані про квартири, в яких номер будинку й квартири збігаються, а також дані про будинки, в яких капітальний ремонт не робили понад 25 років
14	Список абонентів телефонної станції: номер телефону, ПІБ абонента, адреса, заборгованість з абонплати	Вивести дані про абонентів, телефонні номери яких починаються з номера 705, а також про абонентів без заборгованості
15	Список транзисторів: порядковий номер, тип транзистора, матеріал, коефіцієнт передачі по струму	Вивести транзистори германієвого типу, а також список транзисторів, в яких коефіцієнт передачі по струму понад 45 одиниць
16	Список підприємств міста, які виготовляють електронне обладнання: найменування підприємства, вид продукції, кількість продукції за останній за квартал, рентабельність (якщо підприємство рентабельне, то значення цього поля додатне, інакше – від'ємне)	Вивести дані про підприємство, яке виробляє максимальну кількість TV-тюнерів, а також дані про підприємства, які мають від'ємну рентабельність
17	Список нереалізованого товару на складі: найменування товару, кількість, ціна за одиницю товару, дата завезення на склад	Вивести дані про товари, завезені понад рік тому, а також обчислити загальну суму вартості всього товару на складі (кількість * ціна)
18	Список автомобілів в автосалоні: назва, модель, рік випуску, пробіг, ціна	Вивести відомості про усі машини "Тойота" з пробігом менше 20000 км і найти найдешевшмй автомобіль 2014 року випуску

Закінчення табл. 23.1

№ вар.	Вміст бінарного файла	Завдання
19	Список послуг операторів мобільного телефонного зв'язку: назва тарифу, оператор, вартість 1 хв. дзвінків: 1) усередині мережі; 2) на міські номери; 3) на мобільні інших операторів	Вивести дані про тарифи оператора Київстар, а також знайти тариф з найдешевшими дзвінками на мобільні інших операторів
20	Відомості про послуги турагенства: назва туру, країна, кількість днів, вартість туру	Вивести дані про тури до Польщі й визначити найкращий з них (з найменшою вартістю одного дня)
21	Список товару: порядковий номер, назва, виробник, ціна, кількість	Визначити сумарну вартість кожного товару й сумарну вартість усіх товарів, а також вивести дані про товари з ціною понад 1000 грн.
22	Прайс-лист магазину мобільних телефонів: фірма телефону, назва моделі, ціна	Вивести відомості про всі телефони фірми Nokia, а також відомості про телефони з ціною менше 1000 грн.
23	Відомості про квитки на автостанції: пункт призначення, ціна, модель автобусу,час відправлення	Вивести відомості про рейси до Києва, а також знайти найдорожчий рейс
24	Список книг у книжковому магазині: назва книги, автор, рік видання, ціна	Вивести відомості про книги Шевченка, видані після 2000 р., а також визначити найдорожчу книгу
25	Результати футбольного турніру: назва команди, країна, місто, кількість перемог, нічиїх та поразок	Вивести дані про команди з України та обчислити для них сумарні кількості перемог, нічиїх та поразок
26	Результати змагань з легкої атлетики: країна, кількості золотих, срібних та бронзових медалей	Обчислити сумарну кількість медалей для кожної країни. Визначити трійку країн-лідерів
27	Список студентів курсу: порядковий номер, ПІБ, група, середній бал успішності	Визначити студентів з найкращою та найгіршою успішностю, а також вивести студентів тільки з вашої групи
28	Дані про співробітників: табельний номер, ПІБ, посада, стать, рік народження, сімейний стан, кількість дітей	Вивести відомості про холостих чоловіків. Обчислити кількість багатодітних співробітників
29	Результати змагань з легкої атлетики (100 м): ПІБ, стать, країна, результат (час)	Вивести дані про усіх українських спортсменів. Визначити лідера на дистанції на 100 м
30	Список співробітників підприємства: табельний номер, ПІБ, посада, відділ, оклад	Вивести співробітників з відділу реалізації та обчислити їхній сумарний оклад. Визначити найбільший і найменший оклади на підприємстві

Лабораторна робота № 24

Використання dataGridView при роботі з даними файлів

Мета роботи: закріпити практичні навики програмного створення та редагування бінарних файлів за допомогою елемента dataGridView.

Теоретичні відомості

Налаштування dataGridView

Елемент dataGridView на формі можна використовувати не лише для введення-виведення матриць, адже цей елемент має широкий спектр налаштовуваних властивостей, завдяки яким можна задавати різні властивості для різних стовпців елемента (). Цю можливість доволі зручно використовувати для введення-виведення даних бінарних файлів, поля яких мають різні типи даних: числові, рядкові, логічні, списки та ін.

Розглянемо докладніше налаштування dataGridView. Після розміщення такого елемента на формі доцільно почергово створити на ньому необхідну кількість стовпців і при їх створенні відразу задати тип даних, заголовки та інші необхідні значення стовпців, наприклад:

– у властивості HeaderText доцільно вписати текст заголовка відповідного стовпця;

– у властивості ColumnType замість значення за замовчуванням (DataGridViewTextBoxColumn) для текстових і рядкових даних вибрати зі списку одне з можливих значень:

- DataGridViewComboBoxColumn – список з набором можливих текстових значень, для якого у властивості Items доцільно через *Enter* вписати текст можливих значень, наприклад: назви операторів мобільного зв'язку, назви посад співробітників або назви студентських груп;
- DataGridViewCheckBoxColumn – звичайно відповідає логічному типу (значення true та false);
- DataGridViewImageColumn – використовується для виведення зображень;
- DataGridViewButtonColumn – використовується для виведення кнопок у клітинках таблиці;

– властивість AllowUserToAddRows дозволяє додавати нові рядки таблиці при заповненні значень останнього рядка таблиці (за замовчуванням значення true). Значення false забороняє автоматичне додавання нового рядка;

– властивість ColumnHeadersVisible дозволяє приховати заголовки стовпців при значенні false;

– RowHeadersVisible дозволяє приховати заголовки рядків при значенні false;

– властивість RowCount задає кількість рядків;

– властивість ColumnCount задає кількість стовпців;

– властивість Width задає ширину стовпця у пікселях;

– властивість ToolTipText містить текст підказки, яка виводитиметься при наведенні покажчика миші на заголовок стовпця;

– властивість AutoSizeMode дає можливість вибрати режим автоматичного формування ширини того чи іншого стовпця. Приміром, значення AllCells задасть автоматичний підбір ширини стовпця за максимальними вмістом у стовпці;

– властивість ReadOnly вмикає-вимикає дозвіл редагування даних у стовпці;

– властивість SortMode надає можливість задати режим автоматичного сортування даних у стовпці.

– Columns – колекція стовпців;

– Rows – колекція стовпців;

– Font – параметри шрифту.

Приклад програми

Завдання. Створити програмний проект, який реалізує такі завдання:

1) програмне створення бінарного файла і заповнення його двійковими даними: номер мобільного телефону, назву тарифу, назву мобільного оператора, наявність абонплати, вартість одної хвилини розмов, дата підключення;

2) програмний переглядання даних створеного файла;

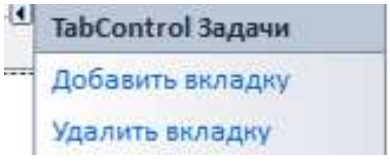
3) можливість редагування та зберігання відредагованого файла;

4) впорядкування (сортування) за зростанням вартості 1 хв.;

5) відбір даних файла про абонентів МТС без абонплати з вартістю 1 хв. розмов понад 50 коп.;

6) відбір абонентів, які користуються послугами мобільного оператора понад три роки.

Розв'язок. Для налаштування форми на ній слід розмістити відповідні командні кнопки та таблиці (елементи dataGridView) для введення-виведення даних файла. Але перед розміщенням на формі кількох dataGridView-ів доцільно створити елемент **tabControl**, який дозволить не захаращувати форму декількома таблицями, а розмістити їх на різних вкладках залежно від розв'язуваної частини завдання: чи то введення даних у файл, чи виведення вмісту всього файла, чи відбір певних даних файла за умовою (запитом). Відтак після розміщення на формі tabControl1 слід утворити на ньому три вкладки, скориставшись командою *Добавить вкладку*, яку можна вибрати з контекстного меню або натиснувши на стрілочку у верхньому правому кутку елемента.



Для першої вкладки створити надпис Заповнення бінарного файла, вписавши його у властивості Text. Для другої та третьої вкладок створити надписи: Виведення вмісту файла та Відбір даних за умовою.

На першій вкладці розмістити кнопку button1 з надписом Записати дані у бінарний файл на перший елемент dataGridView. Після розміщення dataGridView1 на формі слід почергово створити на ньому шість стовпців за кількістю полів у завданні і при їх створенні відразу задати тип даних і заголовки стовпців:

– для 1-го стовпця – у властивості HeaderText вписати Номер телефону;

– для 2-го стовпця – у властивості HeaderText вписати Назва тарифу;

– для 3-го стовпця – у властивості HeaderText вписати Оператор, а для властивості ColumnType замість значення за замовчуванням вибрати зі списку значення DataGridViewComboBoxColumn. Після цього у властивості Items (Items (Коллекция)) вписати через *Enter* назви операторів, наприклад: МТС, Київстар, Life;

– для 4-го стовпця – у властивості HeaderText вписати Наявність абонплати, а для властивості ColumnType замість значення за замовчуванням вибрати зі списку значення DataGridViewCheckBoxColumn;

– для 5-го стовпця – у властивості HeaderText вписати Вартість 1 хв. розмови;

– для 6-го стовпця – у властивості HeaderText вписати Дата підключення.

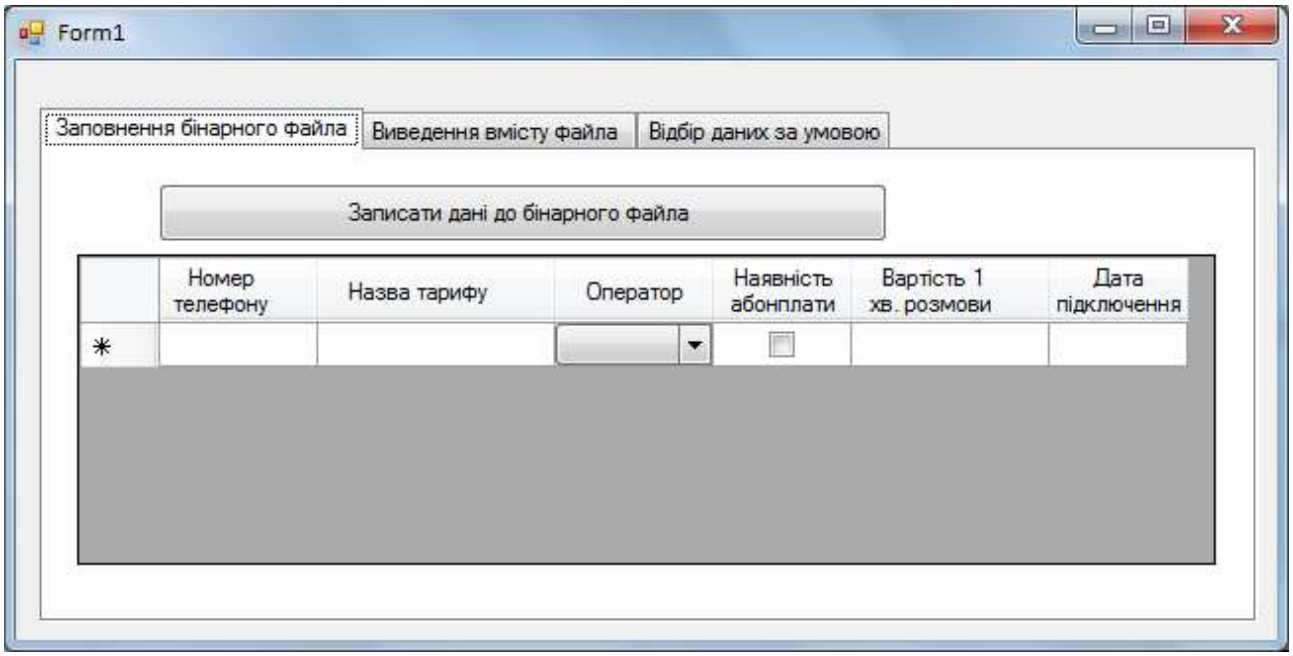
Тепер для всього dataGridView1 доцільно задати стиль заголовка стовпців у колекції властивостей ColumnHeadersDefaultCellStyle для Alignment вибрати значення MiddleCenter. Можна також для кожного зі стовпців задати потрібну ширину у властивості Width, наприклад: для першого і третього стовпців – 80, для четвертого і шостого стовпців – 70.

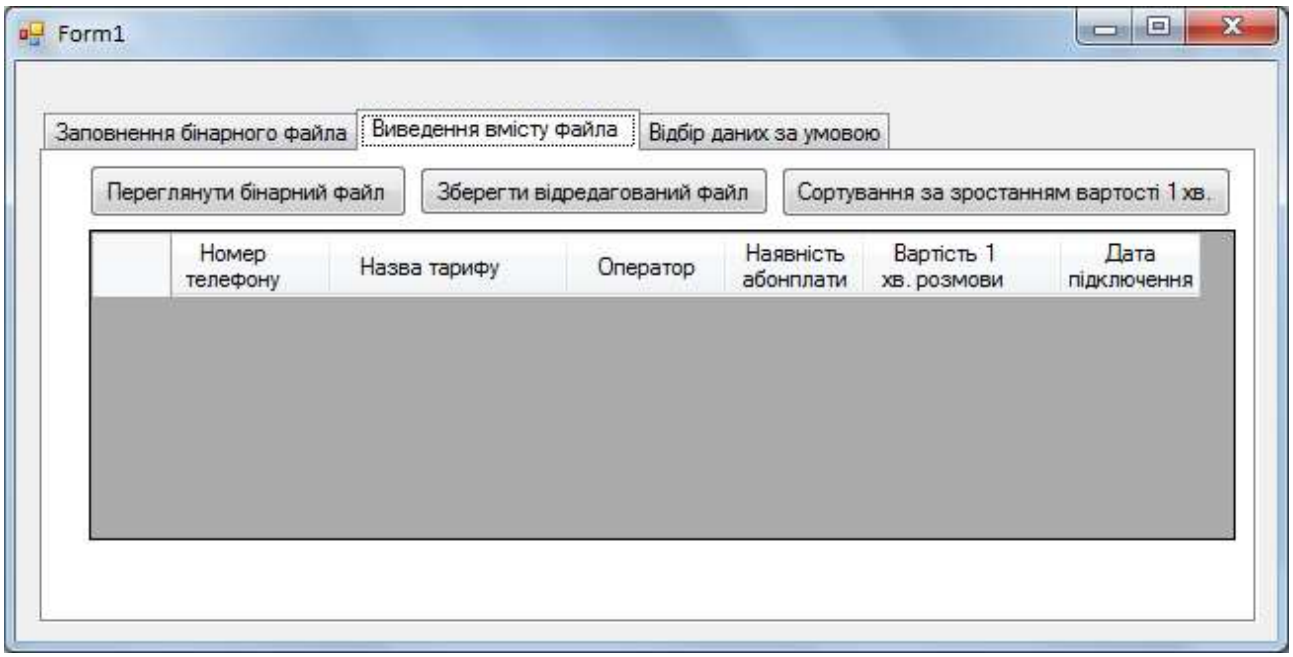
Тепер можна скопіювати dataGridView1 і вставити його копії на другу та третю вкладки, створивши таким чином вже налаштовані dataGridView2 та dataGridView3, для яких лишилось для властивості AllowUserToAddRows встановити значення false.

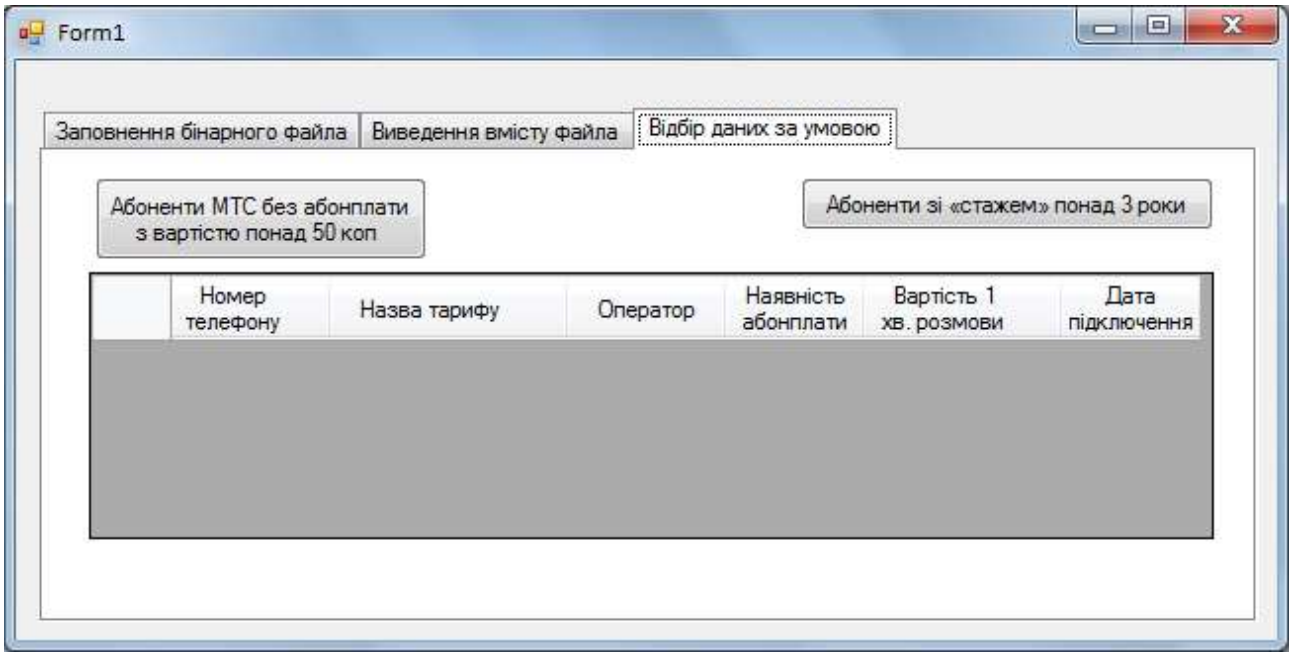
На другій вкладці слід розмістити три кнопки button2, button3 та button4, вписати на них надписи Переглянути бінарний файл, Зберегти відредагований файл та Сортування за зростанням дати підключення.

На третій вкладці розмістити ще дві кнопки button5 та button6, вписати для них текст Абоненти МТС без абонплати з вартістю понад 50 коп. та Абоненти зі «стажем» понад 3 роки.

Після таких налаштувань елементів на формі вона набуде вигляду:







Специфікою файла Form1.h з програмним кодом є наявність значної кількості рядків з командами налаштування елементів на формі, які автоматично створюються у цьому файлі при розміщенні нових елементів на формі та встановленні їх властивостей. Тому, щоб вписати команду для зазначення використання простору імен `using namespace System::IO;` з програмними засобами файлового введення-виведення, слід піднятися на початок файла, віднайти рядок `using namespace System;` і під ним вписати зазначену команду. Після цього спуститися вниз і повернутися до створення програмного коду.

Оскільки ім'я файла буде використовуватись у програмному коді майже всіх кнопок, доцільно для цього оголосити глобальну змінну (наприклад, `fname`), а при створенні форми (функція `Form1_Load`) надати їй значення імені опрацьовуваного бінарного файла. Шаблон функції `Form1_Load` можна утворити подвійним клацанням миші на формі, подібно до того як подвійним клацанням на відповідній кнопці створюються шаблони функцій `button_Click`.

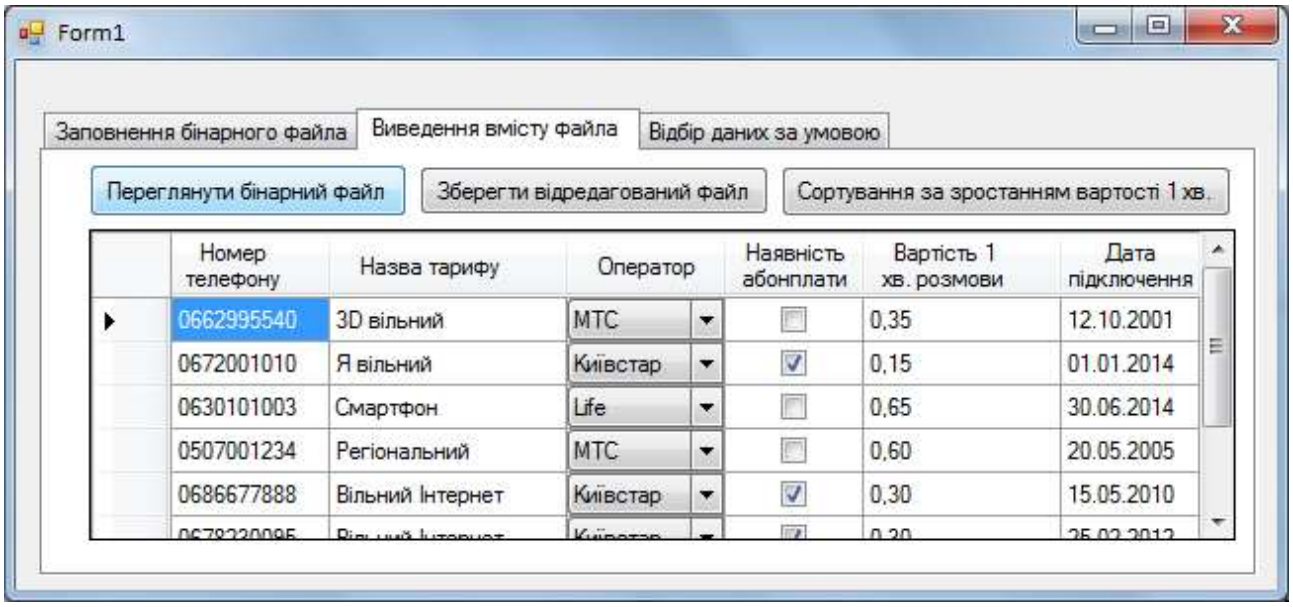
Програмний код:

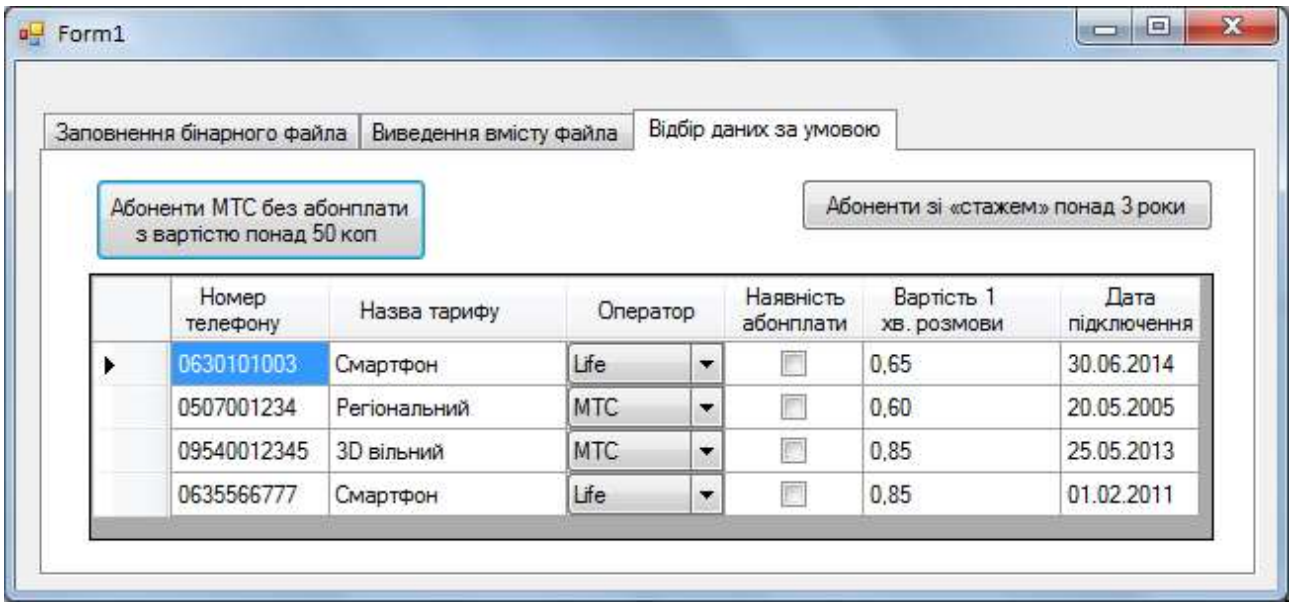
```
. . . . . .
using namespace System;
using namespace System::IO;
. . . . . .
String^ fname; // глобальна змінна - ім'я файла
private: System::Void Form1_Load(System::Object^ sender, System::EventArgs^ e)
{
  fname = "abonent.bin";
}
// Записати дані у бінарний файл
private: System::Void button1_Click(System::Object^sender,System::EventArgs^e)
{
  BinaryWriter^ fb=gcnew BinaryWriter(File::Open(fname,FileMode::OpenOrCreate));
  // або замість цієї команди можна скористатися двома такими
  // FileStream^ phons = gcnew FileStream(fname, FileMode::OpenOrCreate);
  // BinaryWriter^ fb = gcnew BinaryWriter(phons);
  fb->Seek(0, SeekOrigin::End); // перейти у кінець файла для дописування даних
  try
  {
    int k=dataGridView1->Rows->GetRowCount(DataGridViewElementStates::Visible);
    for (int i=0; i < k-1; i++)
    {
      String^ номер = Convert::ToString(dataGridView1[0,i]->Value);
      String^ тариф = Convert::ToString(dataGridView1[1,i]->Value);
      String^ оператор = Convert::ToString(dataGridView1[2,i]->Value);
      bool абонплата = Convert::ToBoolean(dataGridView1[3,i]->Value);
      double vart = Convert::ToDouble(dataGridView1[4,i]->Value);
      String^ d = Convert::ToString(dataGridView1[5,i]->Value);
      fb->Write(номер);       fb->Write(тариф);   fb->Write(оператор);
      fb->Write(абонплата);   fb->Write(vart);    fb->Write(d);
    }
   dataGridView1->Rows->Clear();
  }
  finally
  { fb->Close();
  }
}
// Переглянути бінарний файл
private: System::Void button2_Click(System::Object^sender,System::EventArgs^e)
{
  dataGridView2->Rows->Clear();
  if ( File::Exists(fname) == false )
  { MessageBox::Show("file not exists");  return; }
  BinaryReader^ fb = gcnew BinaryReader(File::OpenRead(fname));
```

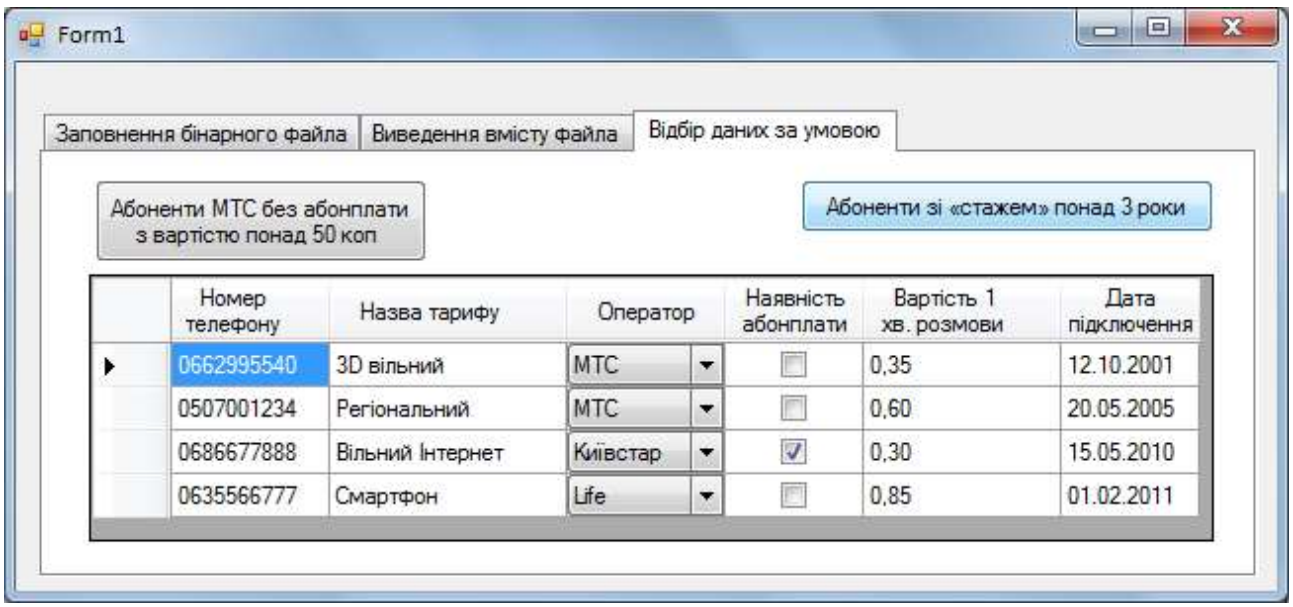
```
  try
  {
    while (fb->BaseStream->Position < fb->BaseStream->Length)
    {
      String^ номер = fb->ReadString();
      String^ тариф = fb->ReadString();
      String^ оператор = fb->ReadString();
      bool абонплата = fb->ReadBoolean();
      double vart = fb->ReadDouble();
      String^ d = fb->ReadString();
      dataGridView2->Rows->Add(номер, тариф, оператор, абонплата,
                                          String::Format("{0:0.00}",vart), d);
  } }
  finally
  { fb->Close(); }
}
// Зберегти відредагований файл
private: System::Void button3_Click(System::Object^sender,System::EventArgs^e)
{
  BinaryWriter^ fb=gcnew BinaryWriter(File::Open(fname,FileMode::Create));
  fb->Seek(0, SeekOrigin::End);
  try
  {
   int kol=dataGridView2->Rows->GetRowCount(DataGridViewElementStates::Visible);
   for (int i = 0; i < kol; i++)
   { String^ номер=Convert::ToString(dataGridView2[0,i]->Value);
     String^ тариф=Convert::ToString(dataGridView2[1,i]->Value);
     String^ оператор=Convert::ToString(dataGridView2[2,i]->Value);
     bool абонплата=Convert::ToBoolean(dataGridView2[3,i]->Value);
     double vart=Convert::ToDouble(dataGridView2[4,i]->Value);
     String^ d=Convert::ToString(dataGridView2[5,i]->Value);
     fb->Write(номер);       fb->Write(тариф);   fb->Write(оператор);
     fb->Write(абонплата);   fb->Write(vart);    fb->Write(d);
   }
  } finally {fb->Close();}
}
// Сортування за зростанням вартості 1 хв. розмов
private: System::Void button4_Click(System::Object^sender,System::EventArgs^e)
{
  button2_Click(button2, e);
  dataGridView2->Sort(dataGridView2->Columns[4],ListSortDirection::Ascending);
  // для сортування за спаданням тип Ascending слід замінити на Descending
 }
// Абоненти МТС без абонплати з вартістю понад 50 коп.
private: System::Void button5_Click(System::Object^sender,System::EventArgs^e)
{
  if ( File::Exists(fname) == false )
  { MessageBox::Show("file not exists");  return;}
```

```
  BinaryReader^ fb = gcnew BinaryReader(File::OpenRead(fname));
  try
  {
    while (fb->BaseStream->Position < fb->BaseStream->Length)
    {
      String^ номер = fb->ReadString();
      String^ тариф = fb->ReadString();
      String^ оператор = fb->ReadString();
      bool абонплата = fb->ReadBoolean();
      double vart = fb->ReadDouble();
      String^ d = fb->ReadString();
      if(!абонплата && vart>=0.5)
       dataGridView3->Rows->Add(номер, тариф, оператор, абонплата,
                                        String::Format("{0:0.00}",vart), d);
    }
  }
  finally
  { fb->Close();
  }
}
// Абоненти зі «стажем» понад 3 роки
private: System::Void button6_Click(System::Object^sender,System::EventArgs^e)
{
  dataGridView3->Rows->Clear(); // Очистити таблицю
  if ( File::Exists(fname) == false )
  { MessageBox::Show("file not exists");  return;}
  BinaryReader^ fb = gcnew BinaryReader(File::OpenRead(fname));
  try
  {
    while (fb->BaseStream->Position < fb->BaseStream->Length)
    { String^ номер = fb->ReadString();
      String^ тариф = fb->ReadString();
      String^ оператор = fb->ReadString();
      bool абонплата = fb->ReadBoolean();
      double vart = fb->ReadDouble();
      String^ d = fb->ReadString();
      DateTime dd = Convert::ToDateTime(d); // Перетворити рядок на дату
      TimeSpan  y = DateTime::Today - dd;   // обчислити різницю дат
      if(y.Days/365.25 >= 3)
        dataGridView3->Rows->Add(номер, тариф, оператор, абонплата,
                                       String::Format("{0:0.00}",vart), d);
    }
  }
  finally
  { fb->Close();
  }
}
```

Результати виконання програми:







Лабораторне завдання

1) У протоколі роботи написати програмний код мовою C++ для:
- створення бінарного файла з ім'ям Вашого прізвища і розширенням `bin`, заповнення файла даними, які вибрати з табл. 24.1 (стовпець *Вміст бінарного файла*);
- програмного перегляду даних створеного файла;
- редагування та зберігання відредагованого файла;
- впорядкування (сортування) даних згідно з параметром, заданим у табл. 24.1;
- відбирання даних файла за певною умовою, заданою у табл. 24.1.

2) Створити на комп'ютері програмні проекти у середовищі Visual C++ для реалізації написаного програмного коду.

Таблиця 24.1

Індивідуальні завдання

№ вар.	Вміст бінарного файла	Параметри сортування	Відбір даних за умовою
1	Заявки на ремонт комп'ютерної техніки: найменування комп'ютера, назва ремонту (список), ПІБ власника, дата взяття на ремонт, кількість днів для ремонтних робіт, вартість ремонту, наявність передплати (так/ні)	За спаданням вартості	Передплачені заявки, для яких кількість днів ремонту є меншою семи
2	Дані про успішність студентів: прізвище та ініціали, дата народження, назва групи (список), оцінки з фізики, математики, історії, наявність стипендії у минулому семестрі (так/ні)	За зростанням оцінок з математики	Студентів, які не здали сесію – мають двійку хоча б з одного предмета – та мали стипендію у минулому семестрі
3	Абоненти телефонної станції: ПІБ абонента, номер телефону, дата встановлення телефону, розмір фіксованої абонплати в місяць (список), наявність пільг (так/ні), сума заборгованості	За спаданням заборгованості	Абонентів без пільг і боргом понад 200 грн.
4	Прайс-лист ноутбуків у магазині: фірма-виробник, тип (список: ноутбук, ультрабук, нетбук, трансформер тощо), діагональ дисплею у дюймах, ціна, дата поставки, наявність сенсорного дисплея (так/ні)	За зростанням ціни	Ноутбуки типу ультрабук з сенсорним дисплеєм
5	Товари в магазині електроніки: код, назва, фірма-виробник (список), рік випуску, вартість, наявність гарантії (так/ні)	За спаданням вартості	Дані про телевізори фірми Samsung з наявною гарантією

Продовження табл. 24.1

№ вар.	Вміст бінарного файла	Параметри сортування	Відбір даних за умовою
6	Список студентів: номер, ПІБ, група (список), дата народження, рейтинг з історії, математики, основ програмування, наявність пропусків занять без поважної причини (так/ні)	В алфавітному порядку прізвищ	Студенти, які претендують на стипендію: мають середній бал понад 75 балів та не мають пропусків занять без поважної причини (так/ні)
7	Прокат автомобілів: державний номер, марка (список), кілометраж пробігу, дата здачі у прокат, сума платні, наявність сигналізації (так/ні)	За спаданням кілометражу	Автомобілі Ford без сигналізації
8	Список робітників підприємства: табельний номер, ПІБ, посада (список), дата народження, наявність вищої освіти (так/ні), розмір зарплатні	За зростанням розміру зарплатні	Співробітників без вищої освіти, вік яких до 40 років
9	Дані в ЖЕКу про мешканців: адреса, назва ділянки чи району (список), ПІБ власника, дата укладення договору про надання послуг, наявність субсидії (так/ні), розмір заборгованості	За спаданням розміру заборгованості	Боржників без субсидії, сума боргу яких понад 500 грн.
10	Список послуг операторів мобільного телефонного зв’язку: номер телефону, назва тарифу (список), дата ініціалізації пакета, наявність підключення до Інтернету (так/ні), баланс рахунку (грн.)	За спаданням номерів телефонів	Дані про абонентів без підключення до Інтернету і балансом рахунку понад 100 грн.
11	Список аварійних будинків у районі міста: вулиця (список), номер будинку, кількість мешканців у будинку, рік зведення, дата постановки на облік, наявність проведення ремонту за останні 20 років (так/ні)	За зростанням кількості мешканців	Вивести відомості про будинки без ремонту за останні 20 років, побудовані до 1941 року, в яких понад 50 мешканців
12	Прайс-лист планшетів у магазині: фірма-виробник (список), модель, діагональ екрана в дюймах, ціна, дата поставки, наявність 3G-модуля (так/ні)	За спаданням значень розміру діагоналі екрана	Планшети Apple з 3G-модулем

Продовження табл. 24.1

№ вар.	Вміст бінарного файла	Параметри сортування	Відбір даних за умовою
13	Список підприємств, які виготовляють електронне обладнання: назва підприємства, дата реєстрації підприємства, вид продукції (список), кількість продукції за останній квартал, рентабельність (так/ні)	За зростанням кількості продукції за останній квартал	Рентабельні підприємства, які виробляють TV-тюнери
14	Список робітників: табельний номер, прізвище та ініціали, дата народження, посада (список), оклад, наявність нагород (так/ні)	За спаданням окладу	Робітники пенсійного віку (понад 60 років), та які мають нагороди
15	Список книг у бібліотеці: інвентарний номер, назва книги, автор, жанр (список), рік видання, ціна, дата інвентаризації, наявність CD/DVD для книги (так/ні)	За зростанням року видання	Книги, у назві яких є "C++", які в комплекті мають CD чи DVD
16	Прайс-лист магазину мобільних телефонів: фірма-виробник (список), модель, діагональ екрана в дюймах, ціна, дата поставки, наявність двох SIM-карт (так/ні)	За спаданням ціни	Телефони Samsung з двома SIM-картами
17	Послуги турагенції: назва туру (список), країна, дата виїзду, кількість днів, вартість туру, наявність нічних переїздів (так/ні)	За зростанням вартості турів	Тури до Польщі без нічних переїздів
18	Список комп'ютерів у комп'ютерному класі: номер, тип ПК (список), тактова частота, оперативна пам'ять, ємність диска, дата встановлення, наявність підключення до мережі Інтернет (так/ні)	За спаданням значень тактової частоти	Підключені до Інтернету ПК, в яких ємність диска не менше 80 Гб
19	Дані про наявність квитків на автостанції: пункт призначення, модель автобусу (список), дата відправлення, час відправлення, ціна, наявність вільних місць (так/ні)	В алфавітному порядку назв пунктів призначення	Дані про рейси до Києва, які відправляються у першій половині дня
20	Каталог ПК: фірма-виробник, тип (список: настільний, моноблок, неттоп, ігровий), ємність RAM, ємність диска, наявність ТВ-тюнера (так/ні)	За зростанням ємності диска	Настільні ПК з ТВ-тюнером
21	Список автомобілів в автосалоні: марка (список), модель, рік випуску, дата реєстрації, пробіг, ціна, наявність гарантійного сервісу протягом року (так/ні)	За спаданням ціни	Автомобілі марки Mazda з гарантійним сервісом

Закінчення табл. 24.1

№ вар.	Вміст бінарного файла	Параметри сортування	Відбір даних за умовою
22	Прайс-лист WIFI-адаптерів у магазині: фірма-виробник (список), модель, швидкість WI-FI у Мбіт/с, ціна, дата поставки, наявність рознімів USB (так/ні)	За спаданням швидкості	WIFI-адаптери фірми Asus з наявністю рознімів USB
23	Анкетні дані про співробітників: табельний номер, ПІБ, посада (список), дата прийому на роботу, сімейний стан, наявність неповнолітніх дітей	В алфавітному порядку ПІБ	Співробітники-чоловіки, які працюють на підприємстві понад 10 років
24	Список книг у книжковому магазині: назва книги, автор, категорія (список), дата надходження, ціна, наявність акційної знижки (так/ні)	За спаданням ціни	Книги категорії "Програмування" з акційною знижкою
25	Список команди "Чорноморець": номер гравця, ПІБ, дата народження, амплуа (список: воротар, захисник тощо), кількість зіграних ігор за минулий сезон, наявність важких травм у цьому сезоні (так/ні)	За спаданням кількості зіграних ігор за минулий сезон	Дані про нападників команди, які не мали важких травм у цьому сезоні
26	Список принтерів для продажу: тип принтера, фірма-виробник (список), марка, ціна принтера, дата поставки, наявність у комплекті запасного картриджа (так/ні)	За зростанням ціни	Принтери фірми HP з ціною до 1000 грн., які мають запасний картридж
27	Список товарів на складі: код, назва, категорія (список), кількість, ціна за одиницю товару, дата завезення на склад, ознака того чи треба товар зберігати в холодильнику (так/ні)	В алфавітному порядку назв товарів	Товари, які зберігаються в холодильнику, завезені понад місяць тому
28	Список студентів курсу: ПІБ, дата народження, група (список), середній бал успішності, проживання у гуртожитку (так/ні)	За зростанням середнього балу	Студентів із середнім балом понад 80 балів, які мешкають у гуртожитку
29	Співробітники підприємства: табельний номер, ПІБ, дата народження, посада, відділ (список), оклад, наявність премії минулого місяця	За спаданням табельних номерів	Робітники з відділу реалізації, які отримали премію
30	Список моніторів: модель, фірма-виробник (список), діагональ у дюймах, можливість підтримки 3D (так/ні), дата поставки	За зростанням розміру діагоналі	Монітори фірми LG без 3D

Лабораторна робота № 25

Відбір даних бінарного файла за умовою з формуванням текстового документа

Мета роботи: закріпити практичні навики програмного відбирання даних бінарного файла та створення на їх основі текстового звітного документа.

Приклад програми

Завдання. У програмному проекті попередньої лабораторної роботи створити можливість програмного відбирання даних за певною умовою та створення на основі відібраних даних текстових звітних документів:

1) абоненти, у назві тарифів яких є слово "вільний";
2) абоненти усіх років, підключені у поточному календарному місяці;
3) абоненти зі стажем понад 5 років.

Розв'язок. Для налаштування форми слід створити для елемента `tabControl1` ще одну вкладку та вписати її надпис `Формування текстового файла`. З попередньої вкладки скопіювати і вставити на щойно створену вкладку елемент `dataGridView`. Крім того, розмістити на цій вкладці три кнопки з надписами 1) `Абоненти, у назві тарифів яких є слово "вільний"`, 2) `Абоненти, підключені у поточному календарному місяці` та 3) `Абоненти зі стажем понад 5 років`. Після цього можна приступити до написання програмного коду для створених кнопок.

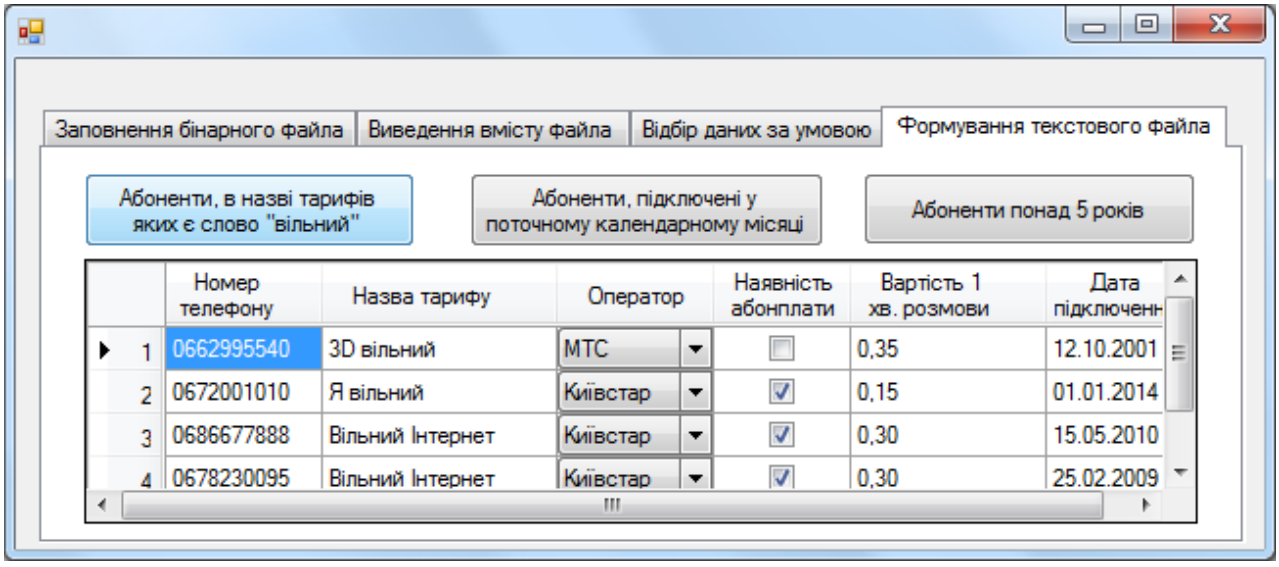
Програмний код:

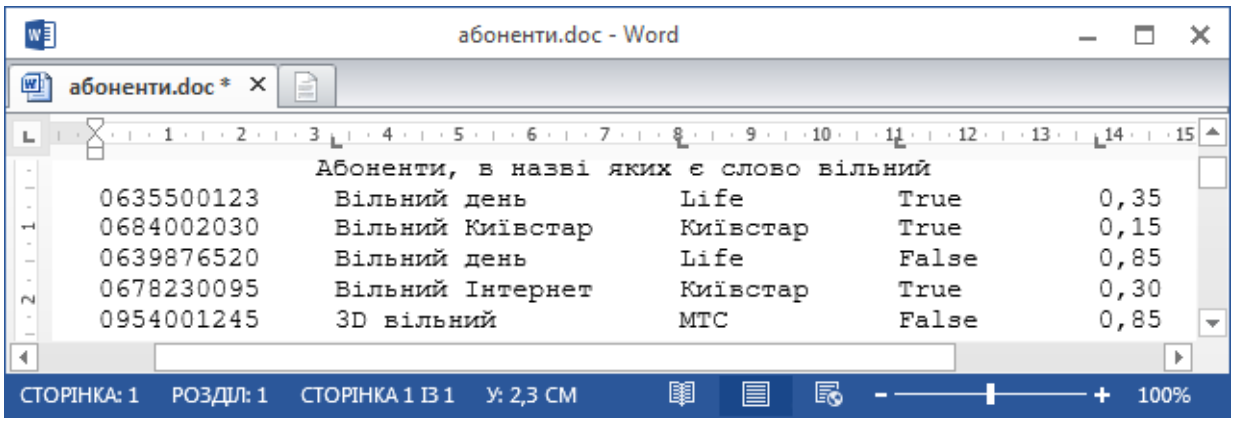
```
// Текстовий файл: абоненти, у назві тарифів яких є слово "вільний"
private: System::Void button7_Click(System::Object^sender,System::EventArgs^e)
{
  int i = 0;
  dataGridView4->Rows->Clear();
  if (File::Exists(fname) == false)
  { MessageBox::Show("file not exists");  return; }
  BinaryReader^ fb = gcnew BinaryReader(File::OpenRead(fname));
  TextWriter^ ft = gcnew StreamWriter(File::OpenWrite("абоненти.doc"));
  try
  {
    ft->WriteLine("\tАбоненти, у назві яких є слово вільний");
    while (fb->BaseStream->Position < fb->BaseStream->Length)
    {
      String^ номер = fb->ReadString();
      String^ тариф = fb->ReadString();
      String^ оператор = fb->ReadString();
      bool абонплата = fb->ReadBoolean();
      double vart = fb->ReadDouble();
      String^ d = fb->ReadString();
```

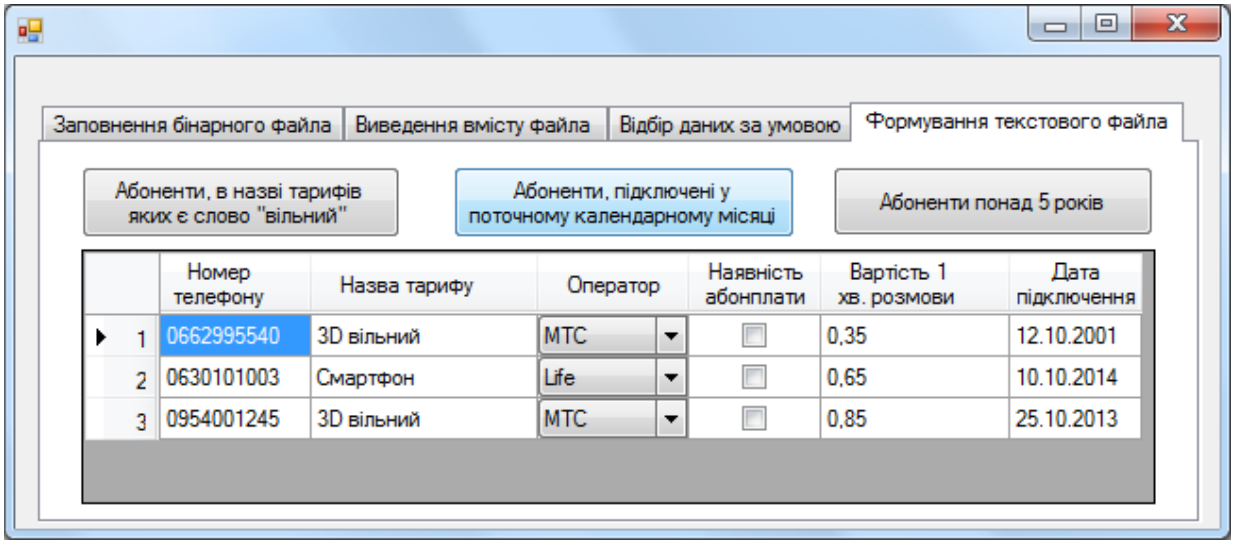
```
      if(тариф->ToLower()->IndexOf("вільний",0) >= 0)
      { dataGridView4->Rows->Add(номер, тариф, оператор, абонплата,
                                        String::Format("{0:0.00}",vart),d);
        dataGridView4->Rows[i]->HeaderCell->Value = (i+1).ToString();
        i++;
        String^ s = номер + "\t" + тариф + "\t" + оператор + "\t" +
             Convert::ToString(абонплата) + "\t" +
             Convert::ToString(String::Format("{0:0.00}",vart) + "\t" + d);
        ft->WriteLine(s);
      }
    }
  } finally {fb->Close(); ft->Close(); }
 }
// Текстовий файл: абоненти всіх років, підключені у поточному календарному місяці
private: System::Void button8_Click(System::Object^sender,System::EventArgs^e)
{
  int i = 0;
  dataGridView4->Rows->Clear();
  if (File::Exists(fname)==false)
  { MessageBox::Show("file not exists");  return; }
  BinaryReader^ fb = gcnew BinaryReader(File::OpenRead(fname));
  TextWriter^ ft = gcnew StreamWriter
                          (File::OpenWrite("Абоненти цього місяця.doc"));
  try
  { ft->WriteLine("Абоненти всіх років,
                  підключені у поточному календарному місяці");
    while (fb->BaseStream->Position < fb->BaseStream->Length)
    { String^ номер = fb->ReadString();
      String^ тариф = fb->ReadString();
      String^ оператор = fb->ReadString();
      bool абонплата = fb->ReadBoolean();
      double vart = fb->ReadDouble();
      String^ d = fb->ReadString();
      DateTime dt = Convert::ToDateTime(d);
      if(dt.Month == DateTime::Today.Month )
      {
       dataGridView4->Rows->Add(номер, тариф, оператор, абонплата,
                                String::Format("{0:0.00}",vart), d);
       dataGridView4->Rows[i]->HeaderCell->Value = (i+1).ToString();
       i++;
       String^ s = номер + "\t" + тариф + "\t" + оператор + "\t" +
             Convert::ToString(абонплата) + "\t" +
             Convert::ToString(String::Format("{0:0.00}",vart) + "\t" + d);
       ft->WriteLine(s);
   } }
  } finally { fb->Close(); ft->Close(); }
}
```

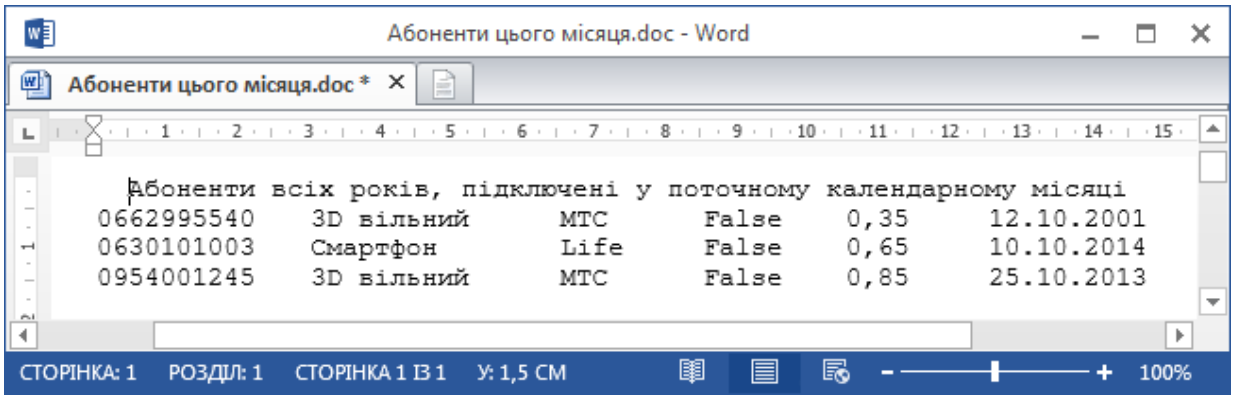
```
// Текстовий файл: абоненти зі стажем понад 5 років
private: System::Void button9_Click(System::Object^sender,System::EventArgs^e)
{ int i = 0;
  dataGridView4->Rows->Clear();
  if (File::Exists(fname) == false)
  { MessageBox::Show("file not exists");  return; }
  BinaryReader^ fb = gcnew BinaryReader(File::OpenRead(fname));
  TextWriter^ ft = gcnew StreamWriter(File::OpenWrite
                                          ("Абоненти понад 5 років.doc"));
  try
  { ft->WriteLine("\tАбоненти зі стажем понад 5 років");
    while (fb->BaseStream->Position < fb->BaseStream->Length)
    { String^ номер = fb->ReadString();
      String^ тариф = fb->ReadString();
      String^ оператор = fb->ReadString();
      bool абонплата = fb->ReadBoolean();
      double vart = fb->ReadDouble();
      String^ d = fb->ReadString();
      DateTime dt=Convert::ToDateTime(d);
      if(DateTime::Today.Year - dt.Year >= 5)
      {  dataGridView4->Rows->Add(номер, тариф, оператор, абонплата,
                                        String::Format("{0:0.00}",vart),d);
         dataGridView4->Rows[i]->HeaderCell->Value = (i+1).ToString();
         i++;
         String^ s = номер + "\t" + тариф + "\t" + оператор + "\t" +
               Convert::ToString(абонплата) + "\t" +
               Convert::ToString(String::Format("{0:0.00}",vart) + "\t" + d);
        ft->WriteLine(s);
      }
    }
  } finally {fb->Close(); ft->Close(); }
}
```

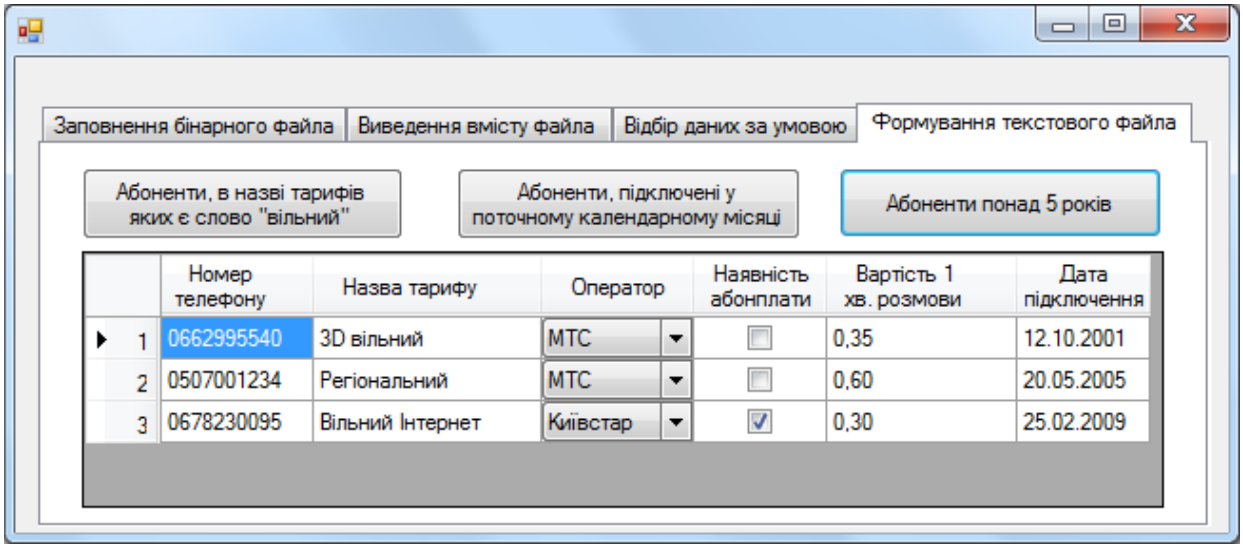
Результати виконання програми:

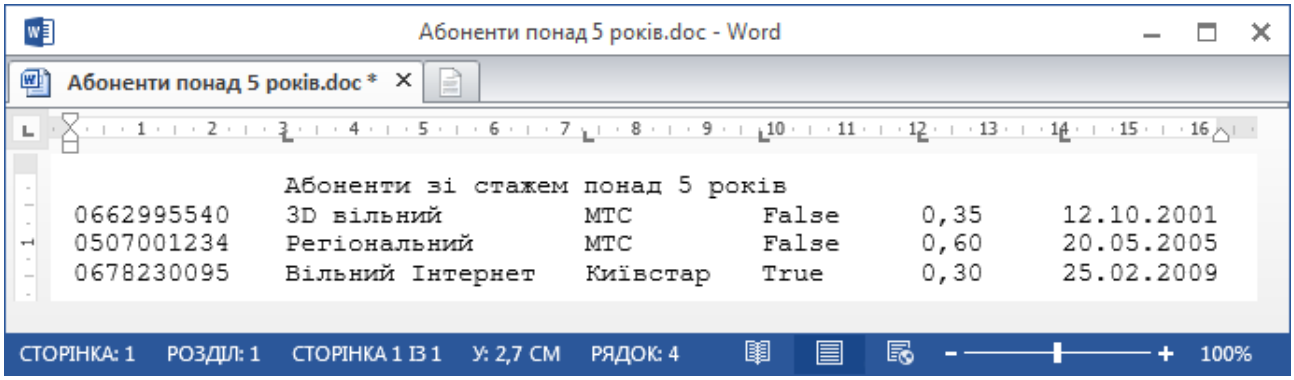



абоненти.doc - Word
абоненти.doc *
Абоненти, в назві яких є слово вільний
0635500123 Вільний день Life True 0,35
0684002030 Вільний Київстар Київстар True 0,15
0639876520 Вільний день Life False 0,85
0678230095 Вільний Інтернет Київстар True 0,30
0954001245 3D вільний МТС False 0,85
СТОРІНКА: 1 РОЗДІЛ: 1 СТОРІНКА 1 ІЗ 1 У: 2,3 СМ 100%


Заповнення бінарного файла
Виведення вмісту файла
Відбір даних за умовою
Формування текстового файла
Абоненти, в назві тарифів яких є слово "вільний"
Абоненти, підключені у поточному календарному місяці
Абоненти понад 5 років
Номер телефону
Назва тарифу
Оператор
Наявність абонплати
Вартість 1 хв. розмови
Дата підключення
1 0662995540 3D вільний МТС 0,35 12.10.2001
2 0630101003 Смартфон Life 0,65 10.10.2014
3 0954001245 3D вільний МТС 0,85 25.10.2013


Абоненти цього місяця.doc - Word
Абоненти цього місяця.doc *
Абоненти всіх років, підключені у поточному календарному місяці
0662995540 3D вільний МТС False 0,35 12.10.2001
0630101003 Смартфон Life False 0,65 10.10.2014
0954001245 3D вільний МТС False 0,85 25.10.2013
СТОРІНКА: 1 РОЗДІЛ: 1 СТОРІНКА 1 ІЗ 1 У: 1,5 СМ 100%


Заповнення бінарного файла
Виведення вмісту файла
Відбір даних за умовою
Формування текстового файла
Абоненти, в назві тарифів яких є слово "вільний"
Абоненти, підключені у поточному календарному місяці
Абоненти понад 5 років
Номер телефону
Назва тарифу
Оператор
Наявність абонплати
Вартість 1 хв. розмови
Дата підключення
1 0662995540 3D вільний МТС 0,35 12.10.2001
2 0507001234 Регіональний МТС 0,60 20.05.2005
3 0678230095 Вільний Інтернет Київстар 0,30 25.02.2009



Лабораторне завдання

1) У протоколі лабораторної роботи написати мовою C++ програмний код для відбирання з бінарного файла попередньої лабораторної роботи даних за умовою, яку вибрати з табл. 25.1, та створити на основі відібраних даних текстовий звітний документ.

2) На комп'ютері у програмному проекті попередньої лабораторної роботи ввести і налагодити розроблений програмний код та впевнитись у правильності його роботи.

Таблиця 25.1

Індивідуальні завдання

№ вар.	Умова на відбір даних з формуванням документа (текстового файла)
1	Заявки на ремонт комп'ютерної техніки, з дати взяття на ремонт яких минуло понад 20 днів
2	Студенти, яким виповнилось 18 років
3	Абоненти, телефони яких встановлені понад три роки тому
4	Ноутбуки, з дати поставки яких минуло понад чотири місяці
5	Товари, випущені понад два роки тому з ціною понад 1000 грн.
6	Студенти, які святкують день народження у цьому місяці
7	Автомобілі, здані у прокат понад тиждень тому
8	Робітники підприємства, які святкують у цьому році ювілей (вік кратний 5)
9	Мешканці, з якими договори укладені понад три роки тому
10	Дані про абонентів, телефони яких ініціалізовані понад рік тому
11	Аварійні будинки, поставлені на облік понад 10 років тому
12	Планшети з діагоналлю 7″, поставлені у цьому місяці
13	Підприємства, зареєстровані понад 6 років тому
14	Робітники, оклади яких більше середнього арифметичного всіх робітників
15	Книги, інвентаризовані у поточному календарному році
16	Телефони, з дати поставки пройшло менше місяця
17	Тури, до дати виїзду яких лишилось менше двох тижнів
18	Комп'ютери, встановлені у комп'ютерному класі понад 4 роки тому
19	Автобусні рейси до Вільнюса, які відправляються у найближчі 10 днів
20	Неттопи фірм Acer або Lenovo

Закінчення табл. 25.1

№ вар.	Умова на відбір даних з формуванням документа (текстового файла)
21	Автомобілі, зареєстровані минулого місяця
22	WIFI-адаптери, з датою поставки у минулому місяці
23	Сімейні співробітники, які мають неповнолітніх дітей
24	Книги, які надійшли за минулий календарний рік
25	Гравці, старше 30 років
26	Принтери, з датою поставки у цьому календарному місяці
27	Товари з ціною менше 70 грн., кількість яких на складі перевищує 1000
28	Неповнолітні студенти (вік менше 18-ти років)
29	Співробітники, яким до пенсії (60 років) лишилося менше 5-ти років
30	Монітори, з датою поставки понад півроку тому

Лабораторна робота № 26

Робота з меню та декількома формами

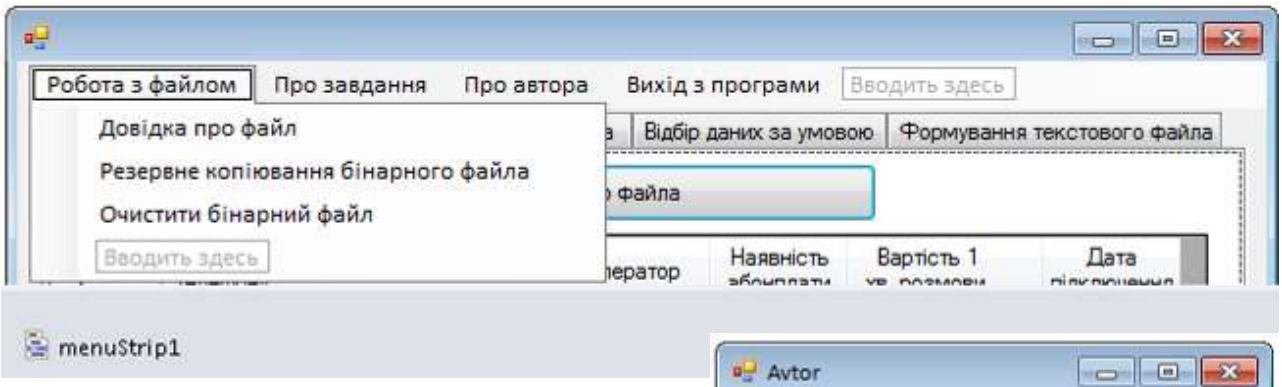
Мета роботи: набути практичних навиків створення програмних проектів з меню та декількома формами.

Приклад програми

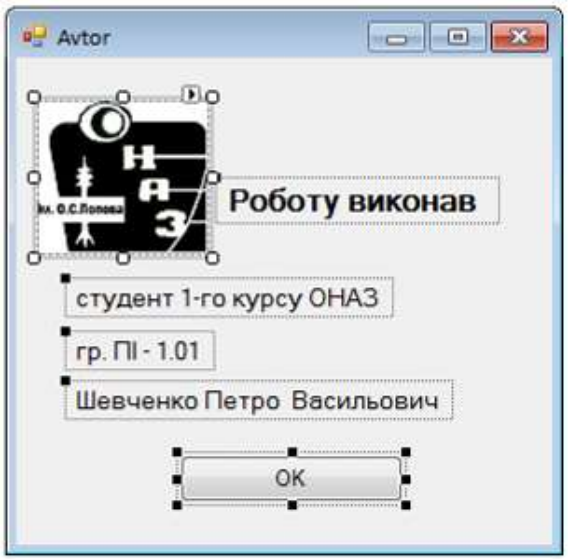
Завдання. У програмному проекті попередньої лабораторної роботи створити три нові форми та програмний код для: 1) виведення інформації про автора, 2) виведення, редагування і форматування параметрів шрифту текстового документа із завданням і 3) форми-заставки. Програмний перехід до створених форм та додаткових функцій для роботи з файлом даних організувати за допомогою меню такої структури:

1) Робота з файлом
   – довідка про файл;
   – резервне копіювання бінарного файла;
   – очистити.
2) Про завдання.
3) Про автора.
4) Вихід з програми.

Розв'язок. На формі програмного проекту, створеного на двох попередніх лабораторних роботах, створити елемент-меню `menuStrip1` та вписати текст написів для конструювання меню такої структури:



Для створення і долучення форми з інформацією про автора слід виконати команду *Проект / Добавить новый элемент / Форма Windows Forms*, ввести ім'я форми – `Avtor` та натиснути кнопку *Добавить*. За допомогою чотирьох елементів `label`, кнопки `button` та `pictureBox1` створити форму на кшталт показаної праворуч. Зображення для елемента `pictureBox1` можна вибрати за допомогою властивості `Image`, а параметри



шрифту для елементів label можна задати за допомогою колекції властивості Font. Після цього слід двічі клацнути по кнопці button1 на цій формі та вписати команду:

```
Close();
```

Тепер треба повернутися до файла Form1.h, піднятися на початок проекту і після команди #pragma once вписати команду підключення щойно створеної форми:

```
#include "Avtor.h"
```

Створення ще двох форм почнемо дещо пізніше, а зараз можна приступити до написання програмного коду для команд створеного меню.

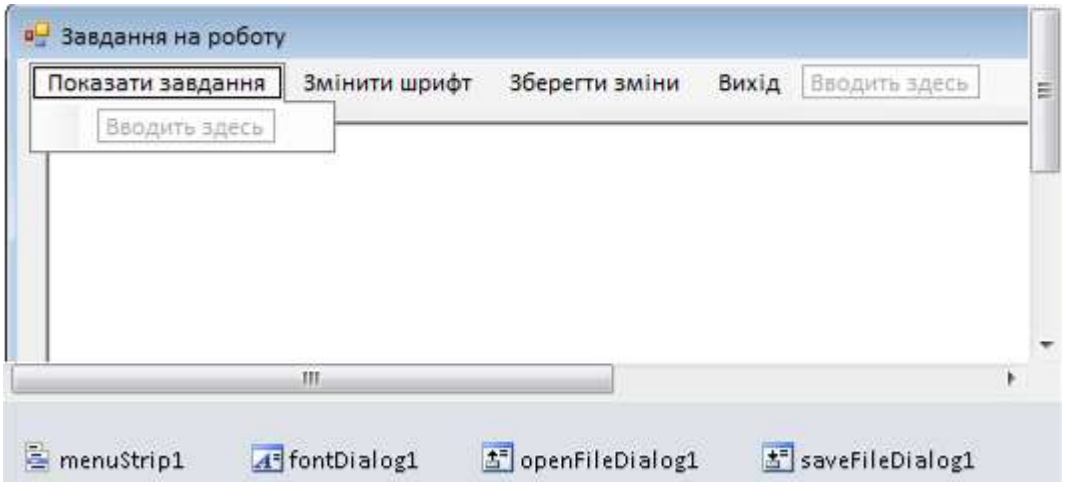
Програмний код для меню:

```
// Довідка про файл
private: System::Void довідкаПроФайлToolStripMenuItem_Click(System::Object^
                                                    sender, System::EventArgs^ e)
{
  if(!File::Exists(fname)) { MessageBox::Show("file not exists");  return; }
  FileInfo^ f=gcnew FileInfo(fname);
  MessageBox::Show("Повне ім'я файла з даними - " +f->FullName +
                   "\nРозмір файла - " + (f->Length).ToString() +
                   " байтів.\nЧас створення - " + f->CreationTime.ToString()+
                   "\nЧас останньої операції дописування даних - " +
                   f->LastWriteTime.ToString());
}
// Резервне копіювання бінарного файла
private: System::Void резервнеКопіюванняToolStripMenuItem_Click
                                    (System::Object^ sender, System::EventArgs^ e)
{ if (!Directory::Exists("Copy"))
    Directory::CreateDirectory("Copy"); // Створити каталог
  FileInfo^ f = gcnew FileInfo(fname);
  f->CopyTo("Copy\\"+fname,true);
}
// Очистити бінарний файл
private: System::Void очиститиФайлToolStripMenuItem_Click
                                    (System::Object^ sender, System::EventArgs^ e)
{
  BinaryWriter^ fb = gcnew BinaryWriter(File::Open(fname, FileMode::Create));
  fb->Close();
  dataGridView2->Rows->Clear();
}
// Про автора
private: System::Void проАвтораToolStripMenuItem_Click
                                    (System::Object^ sender, System::EventArgs^ e)
{
  Avtor^ fr2 = gcnew Avtor();
  fr2->ShowDialog();
}
```

```
// Про завдання
private: System::Void проЗавданняToolStripMenuItem_Click(System::Object^
                                                    sender, System::EventArgs^ e)
{
   Zavdan^ fr3 = gcnew Zavdan();
   fr3->ShowDialog();
}
// Вихід з програми
private: System::Void вихідЗПрограмиToolStripMenuItem_Click
                                    (System::Object^ sender, System::EventArgs^ e)
{
  Close();
}
```

Для створення і долучення нової форми слід виконати команду *Проект / Добавить новый элемент / Форма Windows Forms*, ввести ім'я форми – Zavdan та натиснути кнопку *Добавить*.

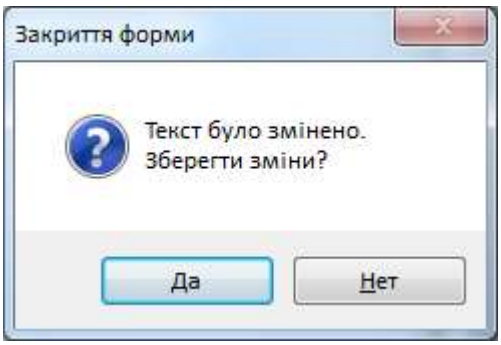
Налаштування форми Zavdan розпочати зі збільшення її ширини та формування надпису на ній Завдання на роботу (властивість Text), після чого розташувати на ній елементи richTextBox1, menuStrip1, fontDialog1, openFileDialog1 та saveFileDialog1. В меню (елемент menuStrip1) цієї форми ввести текст чотирьох команд: 1) Показати завдання; 2) Змінити шрифт; 3) Зберегти зміни та 4) Вихід. Після цього форма Zavdan набуде такого вигляду:



Подвійним клацанням по командам меню створити шаблони функцій та вписати у них такий програмний код:

```
// Показати завдання
private: System::Void показатиЗавданняToolStripMenuItem_Click_1
                                    (System::Object^ sender, System::EventArgs^ e)
{
  openFileDialog1->Filter = "Doc files(*.rtf)|*.rtf";
  if (openFileDialog1->ShowDialog() == System::Windows::Forms::DialogResult::OK)
    richTextBox1->LoadFile(openFileDialog1->FileName);
}
```

```
// Змінити шрифт
private: System::Void змінитиШрифтToolStripMenuItem_Click
                                    (System::Object^ sender, System::EventArgs^ e)
{
  fontDialog1->ShowColor = true;
  fontDialog1->Font = richTextBox1->Font;
  fontDialog1->Color = richTextBox1->ForeColor;
  if (fontDialog1->ShowDialog() !=System::Windows::Forms::DialogResult::Cancel)
  {
    richTextBox1->SelectionFont = fontDialog1->Font;
    richTextBox1->SelectionColor = fontDialog1->Color;
  }
}
// Зберегти зміни
private: System::Void зберегтиЗміниToolStripMenuItem_Click
                                    (System::Object^ sender, System::EventArgs^ e)
{
  saveFileDialog1->Filter = "Doc files(*.rtf)|*.rtf";
  if (saveFileDialog1->ShowDialog() == System::Windows::Forms::DialogResult::OK)
    richTextBox1->SaveFile(saveFileDialog1->FileName);
  richTextBox1->Modified = false;
}
// Вихід
private: System::Void вихідToolStripMenuItem_Click
                                    (System::Object^ sender, System::EventArgs^ e)
 {  // Якщо були зміни, вивести повідомлення з кнопками "Да-Нет"
    // Якщо користувач натиснув "Да", викликати функцію меню "Зберегти зміни"
    if (richTextBox1->Modified)
      if (MessageBox::Show("Текст було змінено.\nЗберегти зміни?",
                           "Закриття форми", MessageBoxButtons::YesNo,
                           MessageBoxIcon::Question) ==
                           System::Windows::Forms::DialogResult::Yes)
       зберегтиЗміниToolStripMenuItem_Click(зберегтиЗміниToolStripMenuItem, e);
    Close();
 }
```



Тепер треба повернутися до файла Form1.h, піднятися на початок проекту і після команди підключення форми Avtor вписати команду підключення щойно створеної форми Zavdan:

```
#include "Zavdan.h"
```

Можна запустити проект та впевнитись у правильності роботи всіх створених функцій, але для того, щоб можна було побачити текст файла із завданням, такий файл слід попередньо створити у текстовому редакторі, наприклад MS Word, записати у нього текст завдання і задати тип файла rtf.

Тепер займемося створенням форми-заставки. Для її створення слід виконати команду *Проект / Добавить новый элемент / Форма Windows Forms*, ввести ім'я форми – Zastavka та натиснути кнопку *Добавить*. Розмістити на формі елементи timer1 та pictureBox1, зображення до якого завантажити за допомого властивості Image.

Подвійним клацанням по формі та по таймеру створити шаблони функцій і вписати у них такий програмний код:

```
private: System::Void Zastavka_Load(System::Object^sender,System::EventArgs^e)
{
  this->timer1->Start();
}
private: System::Void timer1_Tick(System::Object^ sender,System::EventArgs^e)
{
  this->Opacity -= 0.02;//кожної 0,1 секунди прозорість форми зменшуватиметься на 2%
  if (this->Opacity == 0) Close();
}
```

Тепер треба повернутись до файла Form1.h, піднятися на початок проекту і підключити модуль щойно створеної форми Zastavka:

```
#include "Zastavka.h"
```

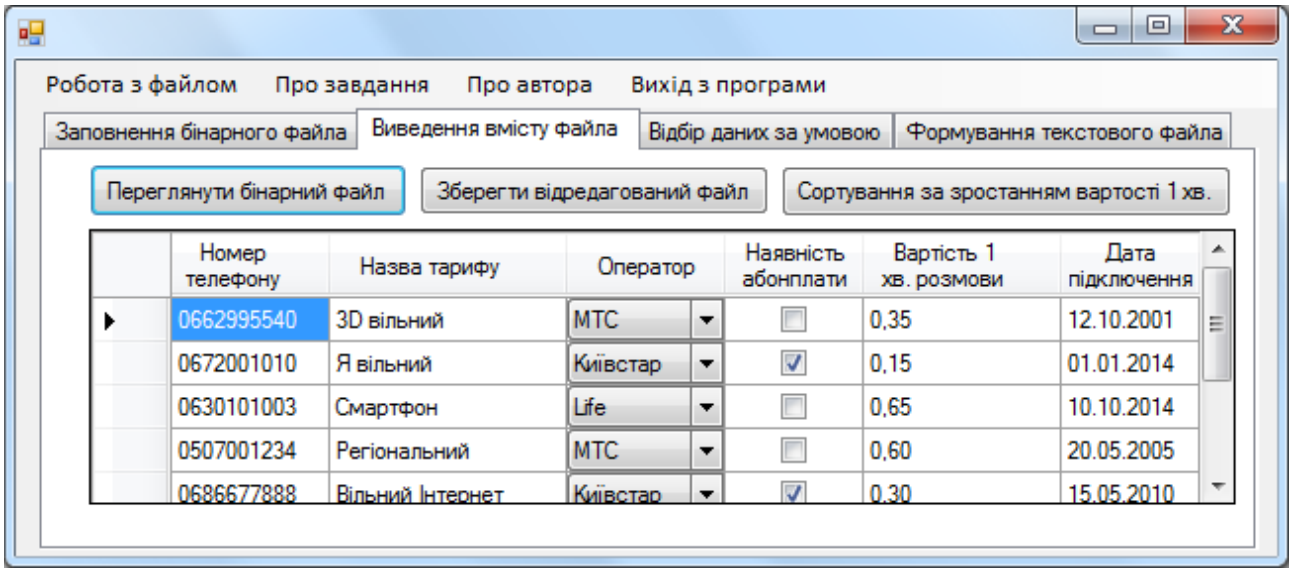
Крім того, слід віднайти функцію Form1_Load та дописати до того оператора, що там є, ще три оператори для запуску форми-заставки:

```
private: System::Void Form1_Load(System::Object^ sender, System::EventArgs^ e)
{
  fname = "abonent.bin";
  this->Hide();
  Zastavka^ fr4 = gcnew Zastavka();
  fr4->ShowDialog();
}
```

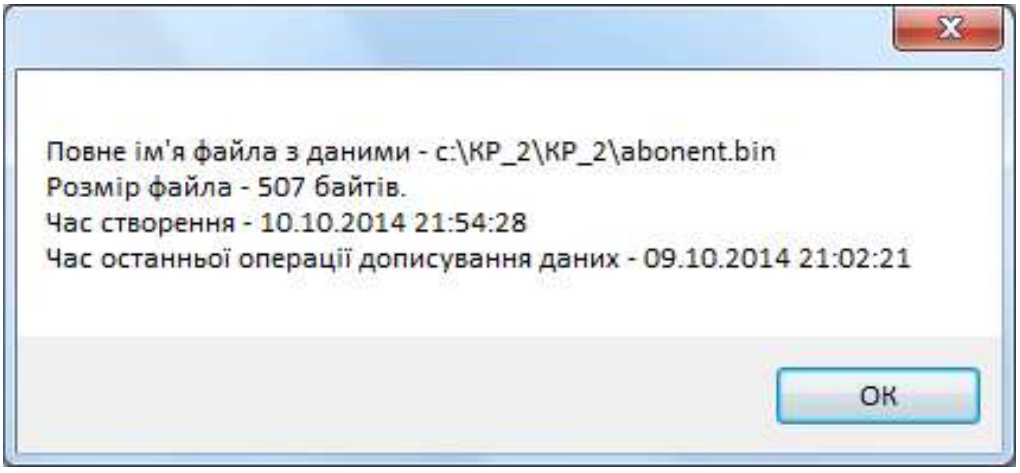
Запустити проект та впевнитись у правильності роботи всіх створених функцій.

Результати виконання програми:

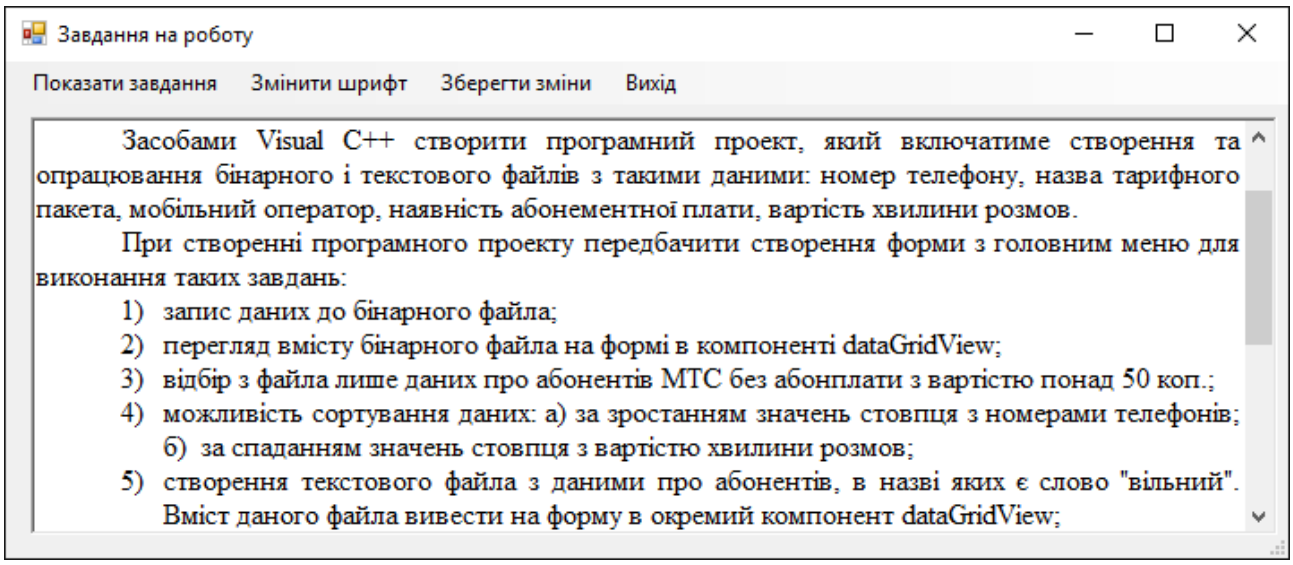
Форма-заставка



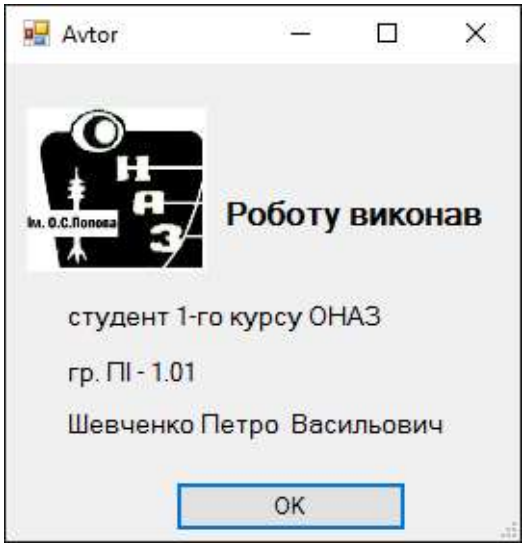
Виведення вмісту файла / Переглянути бінарний файл



Робота з файлом / Довідка про файл



Про завдання



Про автора

Лабораторне завдання

1) У протоколі лабораторної роботи написати мовою C++ програмний код для створення трьох форм: 1) про автора; 2) про завдання та 3) форму-заставку. Крім того, розробити програмний код для виведення, редагування та форматування параметрів шрифту текстового документа із завданням, виведення даних про розмір файла даних, дату його створення і час останнього редагування тощо. Доступ до цих функцій організувати за допомогою меню такої структури:

– Робота з файлом:
  – довідка про файл;
  – резервне копіювання бінарного файла;
  – очистити.
– Про завдання.
– Про автора.
– Вихід з програми.

2) На комп’ютері у проекті попередньої роботи ввести і відлагодити розроблений програмний код та впевнитись у правильності його роботи.

Лабораторна робота № 27

Видалення даних бінарного файла за умовою

Мета роботи: набути практичних навиків програмного видалення даних бінарного файла.

Приклад програми

Завдання. У програмному проекті попередньої лабораторної роботи створити можливість програмного видалення даних за умовою:

1) тарифів "Регіональний";

2) абонентів, підключених менше двох тижнів тому.

Розв'язок. На формі слід створити нові командні кнопки, розмістивши їх на вкладці Відбір даних за умовою. На розміщених кнопках вписати текст Видалити тарифи "Регіональний" та Видалити "новачків", підключених менше двох тижнів тому. Після цього можна розпочати написання програмного коду для створених кнопок.

Структура програмного коду для видалення даних буде такою:

1) при читанні даних з файла у додатковий тимчасовий файл будуть записуватись тільки ті дані, які треба залишити у файлі, а дані, які задовольняють умові видалення, виводитимуться на форму у таблицю (елемент dataGridView3);

2) якщо виявиться, що у файлі не було зазначених в умові даних, про це виведеться відповідне повідомлення, а тимчасовий файл буде знищено. Інакше, буде сформовано діалогове вікно із запитом на підтвердження видалення даних і кнопками "Так" і "Ні";

3) у разі натиснення користувачем кнопки "Так", вихідний файл буде замінений (метод File::Replace) на щойно сформований тимчасовий файл з даними без видалених. Крім того, користувачеві буде запропоновано перейти на другу вкладку, щоб переглянути новий вміст файла;

4) якщо ж користувач натиснув кнопку "Ні", сформований тимчасовий файл буде видалено, тобто вихідний файл залишиться без зміни.

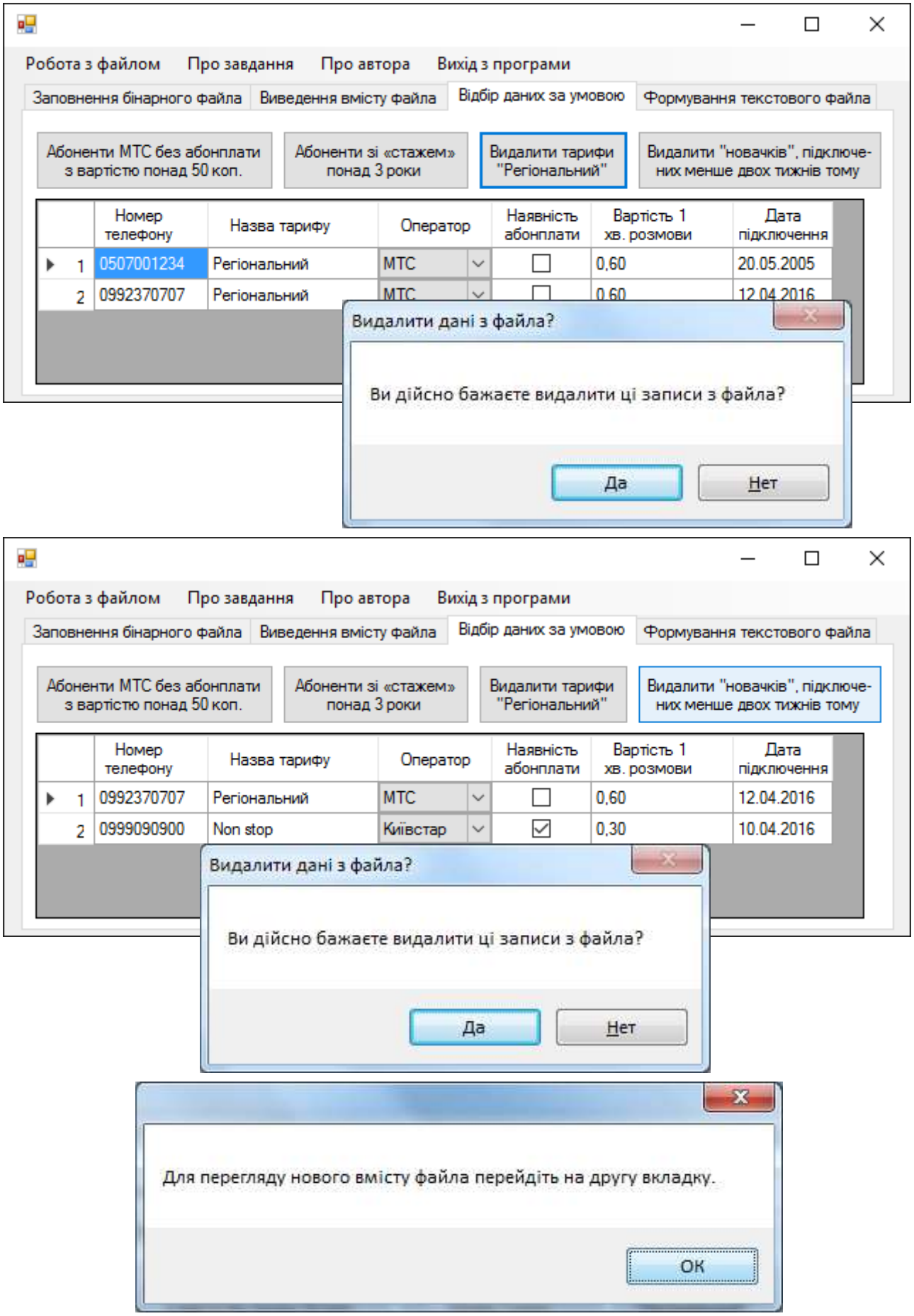
Програмний код:

```
// Видалити тарифи "Регіональний"
private: System::Void button10_Click(System::Object^sender,System::EventArgs^e)
{
   int i = 0;
   dataGridView3->Rows->Clear();
   if (File::Exists(fname) == false)
   {  MessageBox::Show("file not exists");  return;  }
   BinaryReader^ fb = gcnew BinaryReader(File::OpenRead(fname));
   BinaryWriter^ tmp = gcnew BinaryWriter(File::Open("tmp.bin",
                                                     FileMode::Create));
```

```
   try
   {
     while (fb->BaseStream->Position < fb->BaseStream->Length)
     {
      String^ номер = fb->ReadString();
      String^ тариф = fb->ReadString();
      String^ оператор = fb->ReadString();
      bool абонплата = fb->ReadBoolean();
      double vart = fb->ReadDouble();
      String^ d = fb->ReadString();
      if(тариф == "Регіональний")
       {
         dataGridView3->Rows->Add(номер,тариф,оператор,абонплата,
                                          String::Format("{0:0.00}",vart),d);
         dataGridView3->Rows[i]->HeaderCell->Value = (i+1).ToString();
         i++;
       }
       else
       {  tmp->Write(номер);   tmp->Write(тариф);   tmp->Write(оператор);
          tmp->Write(абонплата);   tmp->Write(vart); tmp->Write(d);
       }
     }
     tmp->Close();
     if (!i)
     { MessageBox::Show("Таких записів у файлі не існує");
       File::Delete("tmp.bin");
     }
     else
      if (MessageBox::Show (L"Ви дійсно бажаєте видалити ці записи з файла?",
                            L"Видалити дані з файла?",
                       System::Windows::Forms::MessageBoxButtons::YesNo) ==
                       System::Windows::Forms::DialogResult::Yes)
      {  fb->Close();
         File::Replace("tmp.bin", fname,"Copy\\"+fname);
         dataGridView3->Rows->Clear();
         MessageBox::Show("Для перегляду нового вмісту файла перейдіть
                           на другу вкладку.");
      }
      else File::Delete("tmp.bin");
   } finally {fb->Close(); }
}

// Видалити "новачків", підключених менше двох тижнів тому
private: System::Void button11_Click(System::Object^ sender, System::EventArgs^ e)
{
   int i = 0;
   dataGridView3->Rows->Clear();
```

```
    if(File::Exists(fname) == false)
    {  MessageBox::Show("file not exists");  return;  }
    BinaryReader^ fb = gcnew BinaryReader(File::OpenRead(fname));
    BinaryWriter^ tmp = gcnew BinaryWriter(File::Open("tmp.bin",
                                                      FileMode::Create));
    try
    {
      while (fb->BaseStream->Position < fb->BaseStream->Length)
      {
       String^ номер = fb->ReadString();
       String^ тариф = fb->ReadString();
       String^ оператор = fb->ReadString();
       bool абонплата = fb->ReadBoolean();
       double vart = fb->ReadDouble();
       String^ d = fb->ReadString();
       int days=(DateTime::Today - Convert::ToDateTime(d)).Days;
       if(days < 14)
        {
          dataGridView3->Rows->Add(номер,тариф,оператор,абонплата,
                                          String::Format("{0:0.00}",vart),d);
          dataGridView3->Rows[i]->HeaderCell->Value = (i+1).ToString();
          i++;
        }
        else
        {  tmp->Write(номер);   tmp->Write(тариф);   tmp->Write(оператор);
           tmp->Write(абонплата);   tmp->Write(vart); tmp->Write(d);
        }
      }
      tmp->Close();
      if (!i)
      { MessageBox::Show("Таких записів у файлі не існує");
        File::Delete("tmp.bin");
      }
      else
        if (MessageBox::Show(L"Ви дійсно бажаєте видалити ці записи з файла?",
                             L"Видалити дані з файла?",
                    System::Windows::Forms::MessageBoxButtons::YesNo) ==
                    System::Windows::Forms::DialogResult::Yes)
        {  fb->Close();
           File::Replace("tmp.bin", fname,"Copy\\"+fname);
           dataGridView3->Rows->Clear();
           MessageBox::Show("Для перегляду нового вмісту файла
                             перейдіть на другу вкладку.");
        } else File::Delete("tmp.bin");
    } finally {fb->Close(); }
}
```

Результати виконання програми:



Лабораторне завдання

1) У протоколі лабораторної роботи написати мовою С++ програмний код для видалення даних з бінарного файла попередньої лабораторної роботи даних за умовою, яку вибрати з табл. 27.1.

2) На комп'ютері у програмному проекті попередньої лабораторної роботи ввести і відлагодити розроблений програмний код та впевнитись у правильності його роботи.

Таблиця 27.1

Індивідуальні завдання

№ вар.	Умова на видалення даних
1	Заявки з вартістю ремонтних робіт до 50 грн.
2	Студенти, середній бал яких менший 60
3	Абоненти, борг яких перевищив 500 грн.
4	Ноутбуки, ціна яких менше 2500 грн.
5	Товари, випущені 5 років тому
6	Студенти, які мають пропуски занять без поважної причини
7	Автомобілі, кілометраж пробігу яких понад 100 000 км
8	Робітники підприємства на посаді "оператор"
9	Мешканці, з якими договори укладені в листопаді-місяці
10	Дані про абонентів, баланс рахунку яких менше 1 грн.
11	Аварійні будинки на вулиці Пішонівська
12	Планшети з ціною менше 100 грн.
13	Підприємства, зареєстровані за останні півроку
14	Робітники, день народження яких святкують у найближчі 20 днів
15	Книги, видані до 1950 року
16	Телефони з ціною менше 400 грн.
17	Тури, кількість днів яких понад 18 днів
18	Комп'ютери з розміром оперативної пам'яті до 512 Мб
19	Автобусні рейси, які вже в минулому
20	ПК з ємністю RAM до 1 Гб
21	Автомобілі з пробігом до 5 000 км
22	WIFI-адаптери зі швидкістю до 54 Мбіт/с
23	Співробітники, які прийняті на роботу в цьому році
24	Книги з категорії "Фантастика"
25	Іменинники наступного місяця
26	Принтери з типом "матричний"
27	Товари завезені у лютому-місяці
28	Студентів, які святкують день народження восени
29	Співробітники з окладом до 1500 грн.
30	Монітори з діагоналлю 15 дюймів

Лабораторна робота № 28

Створення звітних документів

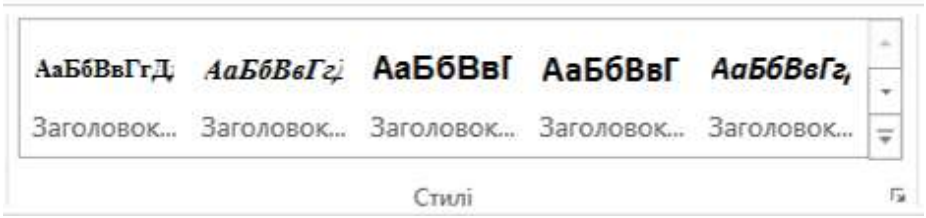
Мета роботи: набути практичних навиків створення і форматування звітних документів у MS Word.

Теоретичні відомості

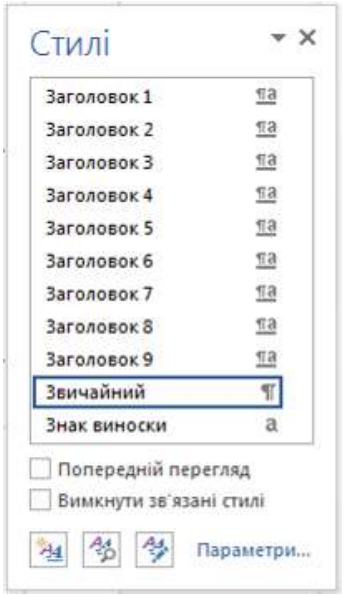
Використання стилів при форматуванні

Існують два способи форматування документів MS Word – **пряме** і **стильове**. Невеликі документи разового використання оформляються першим способом. При оформленні великих документів з різноманітністю стилів абзаців краще виконувати стильове форматування.

Стильове форматування полягає в призначенні спеціальних стилів абзаца і знака. Форматувати текст за допомогою стиля значно швидше, аніж форматувати вручну кожен елемент тексту, оскільки одна команда (стиль) автоматично форматує цілу низку параметрів тексту. Крім того, стильове форматування дозволяє досягти швидкої і зручної уніфікації оформлення всіх документів, які використовуються в певній організації.



Щоб відформатувати фрагмент тексту за допомогою стилю, достатньо виділити його в документі і на стрічці інструментів у вкладці *Основне* у групі *Стилі* вибрати зі списку потрібний стиль.



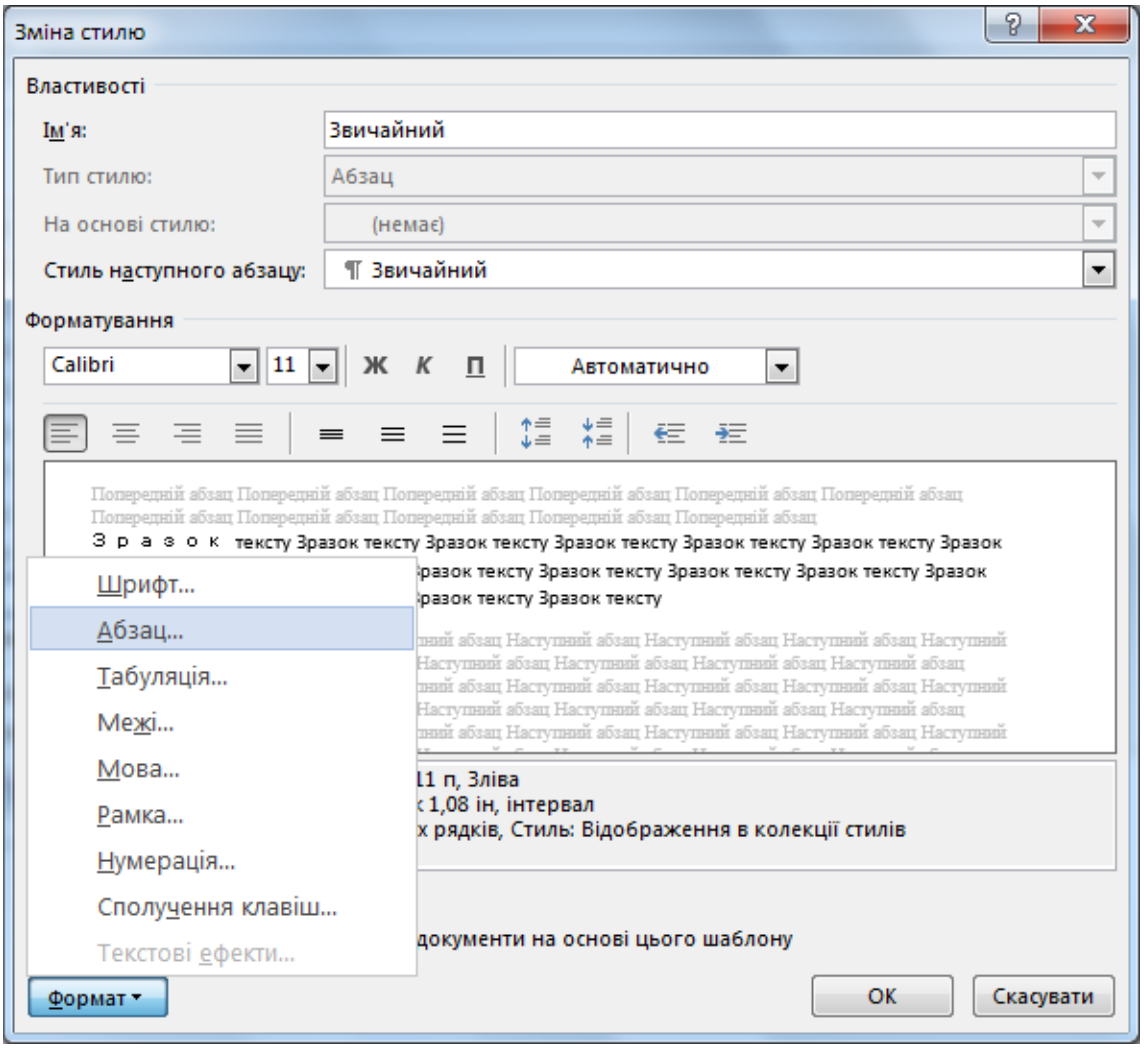
Щоб переглянути список усіх доступних у цьому документі стилів, слід на стрічці інструментів на вкладці *Головна* у групі *Стилі* натиснути на кнопку зі стрілкою (у нижньому правому кутку), після чого відкриється панель *Стилі*.

При форматуванні документа однотипним фрагментам тексту доцільно задавати певний стиль. Наприклад, заголовкам розділів можна призначити стиль *Заголовок 1*, заголовкам підрозділів – стиль *Заголовок 2*, а абзацам основного тексту – стиль *Основний текст* або *Звичайний*. При чому, при зміненні параметрів форматування певного використовуваного стилю, зовнішній вигляд тексту автоматично зміниться по всьому документу.

До речі, форматування заголовків різних рівнів дозволяє переглядати ієрархічну структуру документа, якщо увімкнути *Область переходів* (на вкладці *Вигляд* у групі *Відображення*), а також дозволяє автоматично створювати зміст.

Набір стилів, доступних у процесі створення документа, залежить від шаблона, на якому базується документ. Word дає можливість змінювати, перейменовувати, видаляти і створювати нові стилі для документа чи шаблона.

Для змінення існуючого стилю необхідно встановити курсор в абзац, стиль тексту якого треба змінити. Віднайти у списку всіх доступних стилів стиль цього абзаца (він буде виділений рамкою), клацнути на ньому правою кнопкою миші і вибрати з контекстного меню команду *Змінити*. Це спричинить відкриття діалогового вікна *Зміна стилю*, в якому можна задати нові параметри форматування вибраного стилю.

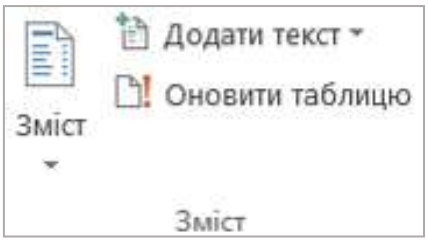


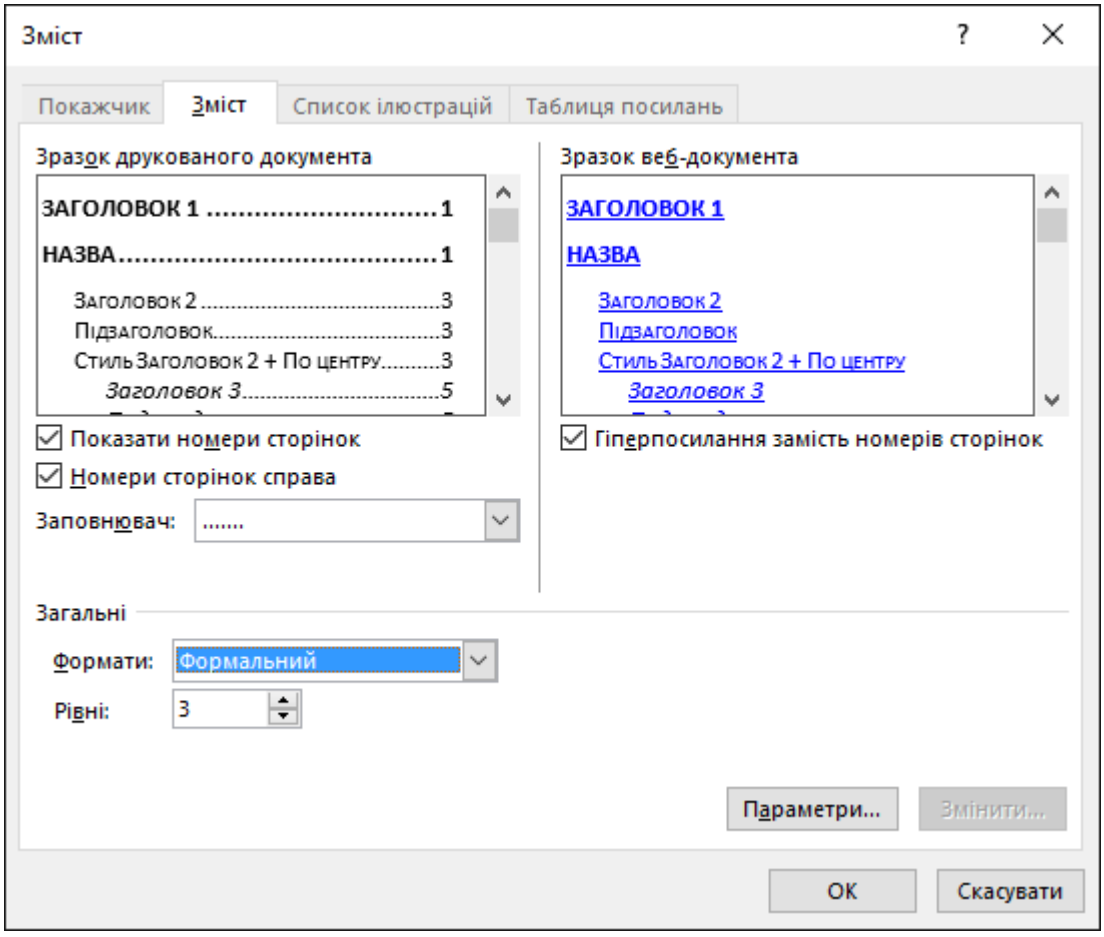
Такі параметри форматування, як відступ першого рядка абзаца, вирівнювання, розміри відступів абзаца можна задати, скориставшись кнопкою *Формат* та вибравши команду *Абзац*. Щоб встановити всі великі букви для стилю *Заголовок 1*, слід скористатись командою *Формат / Шрифт*. Після цього в усіх рядках (приміром, назвах розділів), для яких буде задано стиль *Заголовок 1*, всі літери автоматично перетворяться на великі (прописні).

Створення змісту

У MS Word можна автоматично створювати зміст документа, якщо попередньо були задані стилі заголовків (*Заголовок 1*, *Заголовок 2* тощо), які треба включати в зміст.

Для створення змісту в MS Word 2016 слід на вкладці *Посилання* в групі *Зміст* вибрати необхідний стиль змісту. Додаткові параметри можна задати в діалоговому вікні *Зміст*, яке можна відкрити, натиснувши на стрілочку нижче кнопки *Зміст*, та виконавши команду *Настроюваний зміст*.

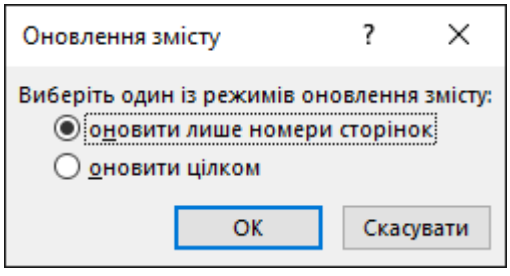




Тут можна вибрати формат змісту, встановити кількість рівнів вкладеності заголовків (за замовчуванням їх 3), задати вид заповнювача вільного місця поміж назвою розділу і номером сторінки та інші параметри змісту, наприклад: параметри шрифту, відступи, інтервали тощо.

Вже після створення змісту можна абсолютно вільно редагувати і форматувати його вміст. Крім того, часто трапляється так, що вже після формування змісту вносяться зміни до назв заголовків або у текст документа, що призводить до зміни кількості сторінок та їхнього вмісту. Автоматично вміст змісту при цьому не змінюється. Для **оновлення змісту** після оновлення вмісту можна скористатися одним з трьох способів:

1) Клацнути правою кнопкою миші у будь-якому місці змісту та вибрати з контекстного меню команду *Оновити поле*. У діалоговому вікні вибрати один з двох варіантів:



а) *оновити лише номери сторінок* – слід вибирати, якщо зміни вносилися тільки в ос-

новний текст, не змінювалися назви заголовків, не додавалися нові розділи, підрозділи тощо;

б) *оновити цілком* – слід вибирати, якщо вносились зміни у назви заголовків, додавалися нові або вилучалися старі заголовки та ін.

2) Клацнути на змісті і натиснути клавішу *F9*. У діалоговому вікні *Оновлення змісту* вибрати один з двох варіантів, розглянутих у попередньому п. 1.

3) Виокремити весь документ (*Ctrl + A*) і натиснути клавішу *F9*. Тоді оновляться всі посилання: зміст, виноски, індекси та ін.

Приклад створення звітного документа

Завдання. Створити звітний документ MS Word з титульною сторінкою, змістом, вступом, чотирма розділами, висновками та списком використаної літератури. Назви розділів і підрозділів можуть бути такими:

1. Теоретичні відомості: інструментарій використовуваний для розв'язання завдання.
2. Програмний код основного модуля з поясненнями.
3. Створення додаткових форм та меню:
   3.1. Функції основного модуля для команд меню.
   3.2. Функції для модуля (форми) "Про автора".
   3.3. Створення модуля (форми) "Завдання".
   3.4. Створення модуля (форми) "Заставка".
4. Здобуті результати у вигляді форм проекту.

При форматуванні звітного документа використовувати стилі *Заголовок 1* – для назв розділів; *Заголовок 2* – для назв підрозділів, формування змісту має бути автоматичним і передбачати можливість оновлення.

Для основного тексту звіту використовувати шрифт: Times New Roman, 14, колір чорний, встановити автоматичне розставлення переносів та задати нумерацію сторінок (крім першої титульної) у нижньому правому кутку.

Послідовність виконання

1) У MS Word створити новий документ з імям *Прізвище_n*, де – номер Вашого варіанта.

2) На першій сторінці створити титулку на кшталт наведеної у додатку А (наприкінці цього файла).

3) За допомогою клавіш *Ctrl + Enter* створити вісім нових сторінок, на кожній з яких ввести такі назви:

Зміст
Вступ
Розділ 1 Теоретичні відомості: інструментарій використовуваний для розв'язання завдання
Розділ 2 Програмний код основного модуля з поясненнями
Розділ 3 Створення додаткових форм та меню
Розділ 4 Вигляд здобутих результатів у вигляді форм проекту
Висновки
Список використаної літератури

4) На сторінці третього розділу сформувати назви таких підрозділів:
3.1. Функції основного модуля для команд меню.
3.2. Функції для модуля (форми) "Про автора".
3.3. Створення модуля (форми) "Завдання".
3.4. Створення модуля (форми) "Заставка".

5) На сторінці зі вступом сформулювати актуальність тематики роботи, тобто розкрити причини важливості розгляду та дослідження вибраної теми, зазначити об'єкт, предмет та мету дослідження, сформулювати завдання, за допомогою якого може бути досягнута мета дослідження. Крім того, у завданні слід зазначити індивідуальний варіант на кшталт такого:

Засобами Visual C++ створити програмний проект, який включатиме створення та опрацювання бінарного і текстового файлів з даними відповідно до індивідуального варіанта. При створенні програмного проекту передбачити виконання таких завдань:

- програмне створення бінарного файла і заповнення його такими даними: номер мобільного телефону, дата підключення (активації) номера, вартість однієї хвилини розмов, назва тарифу, назва мобільного оператора;
- програмний перегляд даних створеного бінарного файла;
- можливість редагування та зберігання відредагованого файла;
- впорядкування (сортування) за зростанням вартості 1 хв. розмов;
- відбір даних файла за умовою: абоненти, які користуються мобільним зв’язком понад три роки;
- можливість програмного відбирання даних за умовою (абоненти, у назві тарифів яких є слово "вільний") та створення на основі відібраних даних текстового звітного документа;
- з метою забезпечення інформаційної частини проекту, а саме: для надання інформації про автора роботи та для ознайомлення із завданням, яке розв’язує проект, створити три додаткові форми: 1) про автора, 2) форму для виведення, редагування та форматування параметрів шрифту текстового документа із завданням і 3) форму-заставку. Перехід до створених форм та додаткових функцій для роботи з файлом даних організувати за допомогою меню.

Крім індивідуального варіанта завдання, у вступі слід вказати структуру роботи, наприклад: "Робота складається зі вступу, чотирьох розділів, висновків та списку використаної літератури. Загальний обсяг роботи – 24 сторінки".

6) Після назви розділу 2 вставити попередньо скопійований програмний код проекту, створеного в лабораторних роботах 24 – 27 (крім програмного коду для меню і додаткових форм). Програмний код для додаткових форм і меню скопіювати і розмістити у відповідних підрозділах розділу 3. Забезпечити докладними коментарями всі ділянки програмного коду.

7) Запустити програмний проект на виконання і за допомогою клавіш *Alt* + *PrtScr* зробити скріншоти форм з результатами на різних етапах виконання програмного проекту після натиснення на різні командні кнопки та команди меню, почергово вставляючи зображення скріншотів у розділ 4 та формуючи відповідні підписи цих рисунків та пояснення до них. Нумерацію рисунків у межах четвертого розділу задати таку: 4.1, 4.2 і т. д.

Підписи рисунків вирівнювати по центру, наприклад:

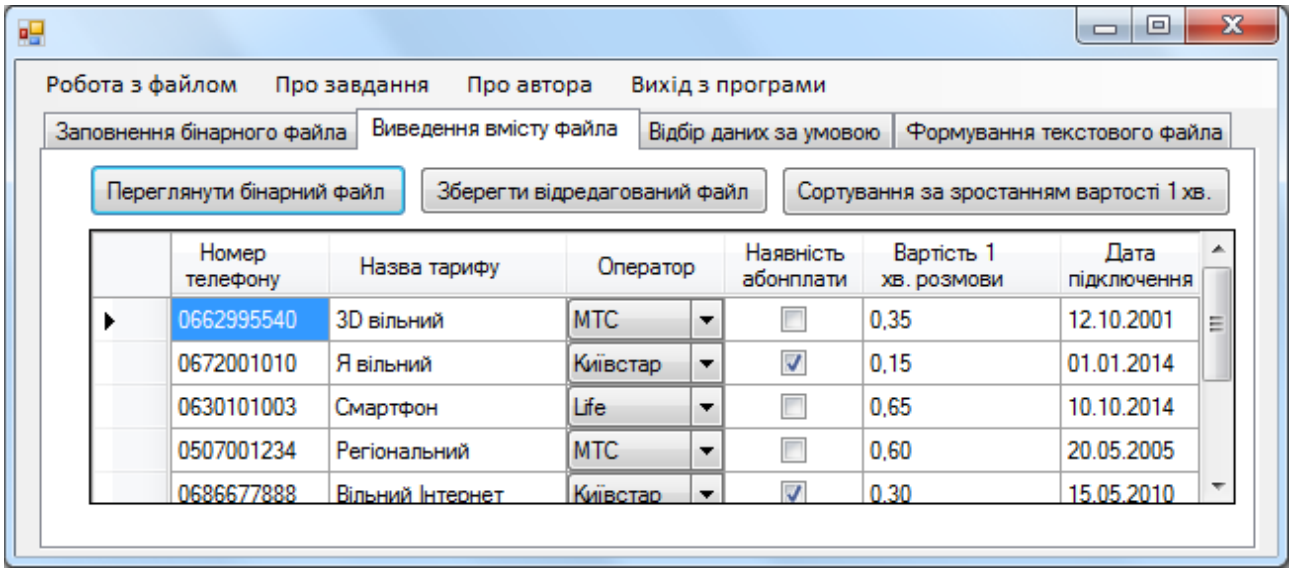


Рис. 4.1. Виведення вмісту файла після натиснення кнопки *Переглянути бінарний файл*

8) У розділі 1 сформувати текст з описом тих засобів, які використовувалися при написанні програмного коду для розв'язання завдання. Цей розділ є теоретичним, у ньому розкривається сутність досліджуваних процесів і використовуваних для розкриття сутності зазначеної роботи методів, може наводитися їх класифікація та різні трактування з позицій різних авторів у науковій літературі.

9) У висновках провести аналіз здобутих результатів та набутих знань і навиків (при цьому заохочується самостійність суджень та оцінок), оцінити повноту та рекомендації щодо можливостей використання матеріалів роботи.

10) Список використаної літератури має бути оформлений за вимогами ДСТУ (ГОСТ) 7.1:2006 "Система стандартів з інформації, бібліотечної та видавничої справи. Бібліографічний запис. Бібліографічний опис. Загальні вимоги та правила складання" і містити 10 – 20 літературних джерел, включаючи посилання на Інтернет-ресурси (див. додаток Б).

11) Коли весь текст буде сформований, виокремити його, крім титульної сторінки, і задати такі параметри:

– форматування шрифту (команда контекстного меню *Шрифт*): Times New Roman, 14, колір чорний;

– форматування абзаца (команда контекстного меню *Абзац*): міжрядковий інтервал – 1; відступ першого рядка – 1,25 см; відступи ліворуч і праворуч – 0; інтервал перед і після – 0;

– вибрати українську мову перевірки орфографії, двічі клацнувши внизу вікна у рядку стану на назві тієї мови, словники якої зараз використовуються, наприклад, замість мови РОСІЙСЬКА вибрати українську.

Щоб встановити автоматичне розставлення переносів, слід на вкладці *Розмітка сторінки* у групі *Параметри сторінки* вибрати команду *Розставлення переносів – Автоматичне*.

Задати поля сторінок (подвійне клацання по боковій лінійці): ліве – 25 мм, верхнє і нижнє – 20 мм, праве – 15 мм.

Задати нумерацію сторінок у нижньому правому кутку, окрім першої титульної сторінки. Для цього на вкладці *Вставлення* у групі *Колонтитули* вибрати команду *Номер сторінки – Внизу сторінки* (справа). Після цього двічі клацнути на номері сторінки, щоб відкрити колонтитул, перейти на вкладку *Знаряддя для колонтитулів – Конструктор*, яка з'явиться, та увімкнути опцію *Інші для першої сторінки*.

12) Встановити курсор на назві першого розділу, на вкладці *Основне* у групі *Стилі* натиснути на кнопку зі стрілкою  (у нижньому правому кутку), для того щоб відкрити панель *Стилі*, з якої вибрати стиль *Заголовок 1*. Такий самий стиль задати для всіх першорядних заголовків (розділи 2, 3, 4, вступ, висновки, список використаної літератури).

Після цього на панелі *Стилі* натиснути праву кнопку миші на стилі *Заголовок 1* та вибрати команду *Змінити*. У діалоговому вікні, яке відкриється натиснути кнопку *Формат* та вибрати команду *Шрифт*, після чого задати такі параметри шрифту: Times New Roman, 16, напівжирний, колір чорний, усі великі букви. Подібним чином виконати команду *Формат – Абзац* та задати параметри: міжрядковий інтервал – 1; відступ першого рядка – 0; відступи ліворуч і праворуч – 0; інтервал перед і після – 12; вирівнювання – по центру.

Після такого змінення параметрів форматування стилю *Заголовок 1* зовнішній вигляд заголовків усіх розділів автоматично зміниться по всьому документу.

13) Стиль *Заголовок 2* застосувати для всіх підрозділів 3.1 – 3.4. Змінити параметри шрифту та абзаца цього стилю на такі:

– шрифт: Times New Roman, 14, напівжирний, колір чорний;

– абзац: міжрядковий інтервал – 1; відступ першого рядка – 1,25; відступи ліворуч і праворуч – 0; інтервал перед – 12 і після – 6; вирівнювання – по лівому краю.

14) Перейти на другу сторінку, встановити курсор у рядок нижче назви ЗМІСТ і на вкладці *Посилання* у групі *Зміст* вибрати необхідний стиль змісту та вставити його.

Після створення змісту виокремити його та задати параметри шрифту та абзаца на кшталт використовуваного форматування в усьому документі.

Оптимальний обсяг звіту – 20 – 25 друкованих сторінок.

15) Зберегти та закрити файл зі звітом та показати його викладачеві для оцінювання.

Додаток А

Міністерство освіти і науки України

Одеська національна академія зв'язку ім. О.С. Попова

Кафедра інформаційних технологій

К У Р С О В А Р О Б О Т А

з дисципліни: "Комп'ютерні технології та програмування"
(назва дисципліни)

на тему: "Створення багатомодульних програмних проектів зі створення та опрацювання даних у файлах засобами Visual C++"

студента 1-го курсу, групи КТ-1.18
напряму підготовки 6.050202
"Комп'ютерні технології та програмування"
Шевченка Костянтина Миколайовича

Керівник доцент, к.т.н. Трофименко О.Г.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Національна шкала _____
Кількість балів: ____ Кількість балів _____
_____ Трофименко О.Г.
(підпис) (прізвище та ініціали)

Одеса 2016

Додаток Б

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. С++. Основи програмування. Теорія та практика: підручник / [О. Г. Трофименко, Ю. В. Прокоп, І. Г. Швайко, Л. М. Буката та ін.] ; за ред. О. Г. Трофименко. – Одеса : Фенікс, 2010. – 544 с.

2. С++. Теорія та практика: навч. посіб. з грифом МОНУ/ [О. Г. Трофименко, Ю. В. Прокоп, І. Г. Швайко, Л. М. Буката та ін.] ; за ред. О. Г. Трофименко. – Одеса : ВЦ ОНАЗ, 2011. – 587 с.

3. Visual C++ .NET: пособие для разработчиков C++ / [А. Корера, С. Фрейзер, С. Джентайл и др.]. – М. : ЛОРИ, 2003. – 398 с.

4. Библиотека классов платформы .NET Framework. [Электронный ресурс]. – Режим доступа: http://msdn.microsoft.com/ru-ru/lib_rary/gg145045 (v=vs.110).aspx. – Название с экрана.

5. Дейтел Х. М. Как программировать на C++; пер с англ. / Х. М. Дейтел, П. Дж. Дейтел. – М. : ООО "Бином-Пресс", 2008. – 1456 с.

6. Довбуш Г. Ф. Visual C++ на примерах / Г.Ф. Довбуш, А.Д. Хомоненко ; под ред. проф. А.Д. Хомоненко. – СПб. : БХВ-Петербург, 2007. – 528 с.

7. Зиборов В. В. MS Visual C++ 2010 в среде .NET. Библиотека программиста / Зиборов В. В. – СПб. : Питер, 2012. – 320 с.

8. Стивен Прата. Язык программирования C++. Лекции и упражнения : учебник ; пер с англ. / Стивен Прата. – СПб.: ООО "ДиаСофтЮП", 2005. – 1104 с.

9. Страуструп Б. Язык программирования C++. Специальное издание ; пер. с англ. / Страуструп Б. – М. : ООО "Бином-Пресс", 2006. – 1104 с.

10. Основи програмування. Базові алгоритми : метод. вказівки для лаб. і практ. робіт / О. Г. Трофименко, Ю. В. Прокоп, І. Г. Швайко, Л. М. Буката. – Ч. 1. – Одеса: ВЦ ОНАЗ ім. О. С. Попова, 2014. – 108 с.

11. Основи програмування. Опрацювання структурованих типів : метод. вказівки для лаб. і практ. робіт / О. Г. Трофименко, Ю. В. Прокоп , І. Г. Швайко, Л. М. Буката. – Ч. 2. – Одеса: ВЦ ОНАЗ ім. О. С. Попова, 2014. – 130 с.

12. Основи програмування. Програмне опрацювання файлів : метод. вказівки для лаб. і практ. робіт / О. Г. Трофименко, Ю. В. Прокоп, І. Г. Швайко, Л. М. Буката. – Ч. 3. – Одеса: ВЦ ОНАЗ ім. О. С. Попова, 2015. – 80 с.

13. Трофименко О. Г. Створення багатомодульних програмних проектів для опрацювання даних у файлах засобами Visual C++: метод. вказівки для виконання курсової роботи з дисципліни "Основи програмування" / Трофименко О. Г., Прокоп Ю. В. – Одеса: ВЦ ОНАЗ ім. О. С. Попова, 2015. – 44 с.

14. Хортон А. Visual C++ 2010: полный курс.; пер. с англ. / Хортон А. – М. : ООО "И.Д. Вильямс", 2011. – 1216 с.

ЗМІСТ