**Виконали:** Свобода Максим, Марусяк Олег

**Група:** КН-308

**Кафедра:** САП

**Дисципліна:** Проектування інформаційних систем

**Перевірив:** Василишин Б.С.

**ЗВІТ**

до лабораторної роботи №2

на тему: “Практичні навички роботи з класами ADO.NET”

**Мета роботи: Ознайомитись зі структурою класів ADO.NET. Здобути практичні навички роботи з класами ADO.NET, що можуть функціонувати без підключення до БД, зокрема DataTable, DataColumn, DataRow, DataSet та інші.**

**Індивідуальне завдання:**

1. Побудувати структуру таблиць DataTable, що відповідає структурі індивідуального завдання.
2. Об’єднати створені таблиці в об'єкті DataSet реалізувавши зв'язки згідно з індивідуальним завданням.
3. Заповнити даними створену структуру(мінімум 5 рядків в кожній таблиці). Вивести дані з таблиць користувачу.
4. За допомогою DataView реалізувати сортування та фільтрування однієї з таблиць та вивести дані з цього DataView користувачу.
5. Внести зміни в кожну з таблиць(додати, видалити і модифікувати, як мінімум один рядок).
6. Сформувати новий DataSet, який міститиме тільки змінені дані(додані, модифіковані, видалені) та відобразити його користувачу при цьому відобразити інформацію тип змін, що відбулись в кожному з рядків(додавання/модифікація/видалення).

**Завдання 1:** Побудувати структуру таблиць DataTable

Об’єктом нашої бази даних є лікарня. Отже, створюємо таблиці, оголошуємо відповідні атрибути та заповнюємо їх даними наступним чином:

public DataTable dt; // Це для таблиць

public DataColumn dc; // це для атрибутів

public DataRow dr; // Це для ініціалізації таблиць

public static DataSet changesDS;

public DataTable GetEquipmentTable()

{

dt = new DataTable("Спорядження"); // роблю таблицю

dc = new DataColumn("ID", typeof(int)); //додаю перший атрибут

dt.Columns.Add(dc);

dt.PrimaryKey = new DataColumn[] { dc }; // роблю його первинним ключем

dc = new DataColumn("Name", typeof(string));

dt.Columns.Add(dc);

dc = new DataColumn("Quantity", typeof(int));

dt.Columns.Add(dc);

dc = new DataColumn("Manufacturer", typeof(string));

dt.Columns.Add(dc);

dr = dt.NewRow(); // Додаю інші атрибути

dr[0] = 1;

dr[1] = "Шприц";

dr[2] = 100;

dr[3] = "МедПласт";

dt.Rows.Add(dr); // Приклад заповнення таблиці

dr = dt.NewRow();

dr[0] = 2;

dr[1] = "Томограф МРТ";

dr[2] = 1;

dr[3] = "MR United";

dt.Rows.Add(dr);

dt.Rows.Add(3, "Пульсоксиметр", 10, "OSD"); // Другий приклад заповнення таблиці

dt.Rows.Add(4, "Дефибрилятор", 5, "Praimed");

dt.Rows.Add(5, "Стетоскоп", 15, "WBI");

dt.Rows.Add(6, "Стоматологічна установка", 1, "MedMax");

dt.Rows.Add(7, "Візіограф", 1, "DenTests");

dt.Rows.Add(8, "Апарат для електроодонтодіагностики", 1, "DenTests");

dt.Rows.Add(9, "Апарат УЗД", 1, "UltraSFX");

dt.Rows.Add(10, "Кардіограф", 1, "HB Medical");

return dt;

}

public DataTable GetMedicineTable()

{

DataTable dt = new DataTable("Ліки");

DataColumn dc = new DataColumn("ID", typeof(int));

dt.Columns.Add(dc);

dt.PrimaryKey = new DataColumn[] { dc };

dc = new DataColumn("Name", typeof(string));

dt.Columns.Add(dc);

dc = new DataColumn("Producer", typeof(string));

dt.Columns.Add(dc);

dc = new DataColumn("Created Date", typeof(DateOnly));

dt.Columns.Add(dc);

dc = new DataColumn("Expiration Date", typeof(DateOnly));

dt.Columns.Add(dc);

DateOnly date1 = new DateOnly(2023, 05, 05); DateOnly date2 = new DateOnly(2026, 05, 05);

dt.Rows.Add(1, "Фітотроп", "MedMax", date1, date2);

date1 = new DateOnly(2023, 02, 02); date2 = new DateOnly(2026, 02, 02);

dt.Rows.Add(2, "Кромофарм", "ФармаК", date1, date2);

date1 = new DateOnly(2021, 12, 25); date2 = new DateOnly(2023, 12, 25);

dt.Rows.Add(3, "Тантум Верде", "А.К.Р.А.Ф", date1, date2);

date1 = new DateOnly(2021, 08, 11); date2 = new DateOnly(2023, 08, 11);

dt.Rows.Add(4, "Називін", "Dr.Reddy's", date1, date2);

date1 = new DateOnly(2022, 12, 10); date2 = new DateOnly(2024, 12, 10);

dt.Rows.Add(5, "Біле Вугілля", "Борщагівський ХФЗ", date1, date2);

dt.Rows.Add(6, "Панкреатин", "Технолог", new DateOnly(2020, 11, 09), new DateOnly(2026, 11, 09));

dt.Rows.Add(7, "Лідоксан Ментол", "Sandoz", new DateOnly(2019, 06, 15), new DateOnly(2024, 06, 15));

dt.Rows.Add(8, "Ко-Пренелія", "Arterium", new DateOnly(2022, 11, 10), new DateOnly(2026, 11, 10));

dt.Rows.Add(9, "Plavix", "Sanofi-Aventis", new DateOnly(2021, 11, 29), new DateOnly(2025, 11, 29));

dt.Rows.Add(10, "Спрей-Бар\'єр", "Імідж", new DateOnly(2020, 06, 06), new DateOnly(2024, 05, 05));

return dt;

}

public DataTable GetPatientTable()

{

dt = new DataTable("Пацієнти");

dc = new DataColumn("ID", typeof(int));

dt.Columns.Add(dc);

dt.PrimaryKey = new DataColumn[] { dc };

dc = new DataColumn("FirstName", typeof(string));

dt.Columns.Add(dc);

dc = new DataColumn("LastName", typeof(string));

dt.Columns.Add(dc);

dc = new DataColumn("MiddleName", typeof(string));

dt.Columns.Add(dc);

dc = new DataColumn("Age", typeof(int));

dt.Columns.Add(dc);

dc = new DataColumn("Disease", typeof(string));

dt.Columns.Add(dc);

dc = new DataColumn("ArrivalDate", typeof(DateOnly));

dt.Columns.Add(dc);

dc = new DataColumn("DischargeDate", typeof(DateOnly));

dt.Columns.Add(dc);

dc = new DataColumn("Ward", typeof(string));

dt.Columns.Add(dc);

DateOnly date1 = new DateOnly(2023, 08, 10); DateOnly date2 = new DateOnly(2023, 08, 24);

dt.Rows.Add(1, "Олена", "Іванов", "Михайлівна", 21, "Грип", date1, date2, "Отоларингологічне відділення");

date1 = new DateOnly(2023, 07, 12); date2 = new DateOnly(2026, 08, 13);

dt.Rows.Add(2, "Андрій", "Петров", "Володимирович", 28, "Корона-вірус", date1, date2, "Отоларингологічне відділення");

date1 = new DateOnly(2023, 09, 02);

dt.Rows.Add(3, "Марія", "Сидоренко", "Петрівна", 19, "Сколіоз", date1, null, "Ортопедичне відділення");

date1 = new DateOnly(2023, 08, 25);

dt.Rows.Add(4, "Володимир", "Ковальчук", "Ігорович", 15, "Перелом", date1, null, "Ортопедичне відділення");

date1 = new DateOnly(2023, 07, 15); date2 = new DateOnly(2023, 08, 01);

dt.Rows.Add(5, "Лариса", "Григоренко", "Олександрівна", 25, "Цироз печінки", date1, date2, "Гастроентерологічне відділення");

dt.Rows.Add(6, "Сергій", "Шевченко", "Іванович", 36, "Апендицит", new DateOnly(2023, 06, 03), new DateOnly(2023, 06, 06), "Хірургічне відділення");

dt.Rows.Add(7, "Тетяна", "Мельник", "Василівна", 67, "Мікроінфаркт", new DateOnly(2023, 07, 07), new DateOnly(2023, 07, 21), "Кардіологічне відділення");

dt.Rows.Add(8, "Олександр", "Козак", "Федорович", 85, "Перелом", new DateOnly(2023, 09, 01), null, "Ортопедичне відділення");

dt.Rows.Add(9, "Ірина", "Павленко", "Юріївна", 43, "Асма", new DateOnly(2023, 07, 27), new DateOnly(2023, 08, 15), "Отоларингологічне відділення");

dt.Rows.Add(10, "Михайло", "Ковач", "Вікторович", 51, "Грип", new DateOnly(2023, 09, 06), null, "Отоларингологічне відділення");

return dt;

}

public DataTable GetStaffTable()

{

dt = new DataTable("Персонал");

dc = new DataColumn("ID", typeof(int));

dt.Columns.Add(dc);

dt.PrimaryKey = new DataColumn[] { dc };

dt.Columns.Add("FirstName", typeof(string));

dt.Columns.Add("LastName", typeof(string));

dt.Columns.Add("MiddleName", typeof(string));

dt.Columns.Add("Age", typeof(int));

dt.Columns.Add("Position", typeof(string));

dt.Rows.Add(1, "Михайло", "Зубенко", "Петрович", 45, "Хірург");

dt.Rows.Add(2, "Любомир", "Недзь", "Сергійович", 30, "Сімейний лікар");

dt.Rows.Add(3, "Оксана", "Балдур", "Михайлівна", 20, "Медсестра");

dt.Rows.Add(4, "Михайло", "Щурич", "Ігорович", 30, "Оториноларинголог");

dt.Rows.Add(5, "Софія", "Чупляк", "Романівна", 61, "Терапевт");

dt.Rows.Add(6, "Ольга", "Шабель", "Петрівна", 38, "Масажист");

dt.Rows.Add(7, "Андрій", "Зладо", "Пилипович", 27, "Педіатр");

dt.Rows.Add(8, "Олександр", "Вольський", "Любомирович", 23, "Ревматолог");

dt.Rows.Add(9, "Минай", "Мазайло", "Олегович", 53, "Ендокринолог");

dt.Rows.Add(10, "Катерина", "Ватруш", "Вікторівна", 31, "Уролог");

return dt;

}

public DataTable GetProceduresTable()

{

DataTable dt = new DataTable("Процедури");

dc = new DataColumn("ProcedureID", typeof(int));

dt.Columns.Add(dc);

dt.PrimaryKey = new DataColumn[] { dc };

dt.Columns.Add("PatientID", typeof(int));

dt.Columns.Add("EquipmentID", typeof(int));

dt.Columns.Add("StaffID", typeof(int));

dt.Columns.Add("MedID", typeof(int));

dt.Columns.Add("Name", typeof(string));

dt.Columns.Add("Price", typeof(int));

dt.Rows.Add(1, 3, 1, 1, 1, "Щоранкові процедури", 50);

dt.Rows.Add(2,2,5,2,5, "Введення противірусної сироватки",75);

dt.Rows.Add(3,1,5,4,2, "Встановлення катетора",100);

dt.Rows.Add(4,5,5,2,3, "Щоранкові процедури",50);

dt.Rows.Add(5,4,1,1,5, "Взяття крові з пальця",50);

dt.Rows.Add(6, 6, 9, 1, 5, "Проведення УЗД", 550);

dt.Rows.Add(7, 3, 2, 3, 9, "Проведення МРТ для хребта", 1800);

dt.Rows.Add(8, 7, 4, 7, 8, "Екстрена реанімація", 250);

dt.Rows.Add(9, 2, 3, 3, 2, "Перевірка цукру в крові", 50);

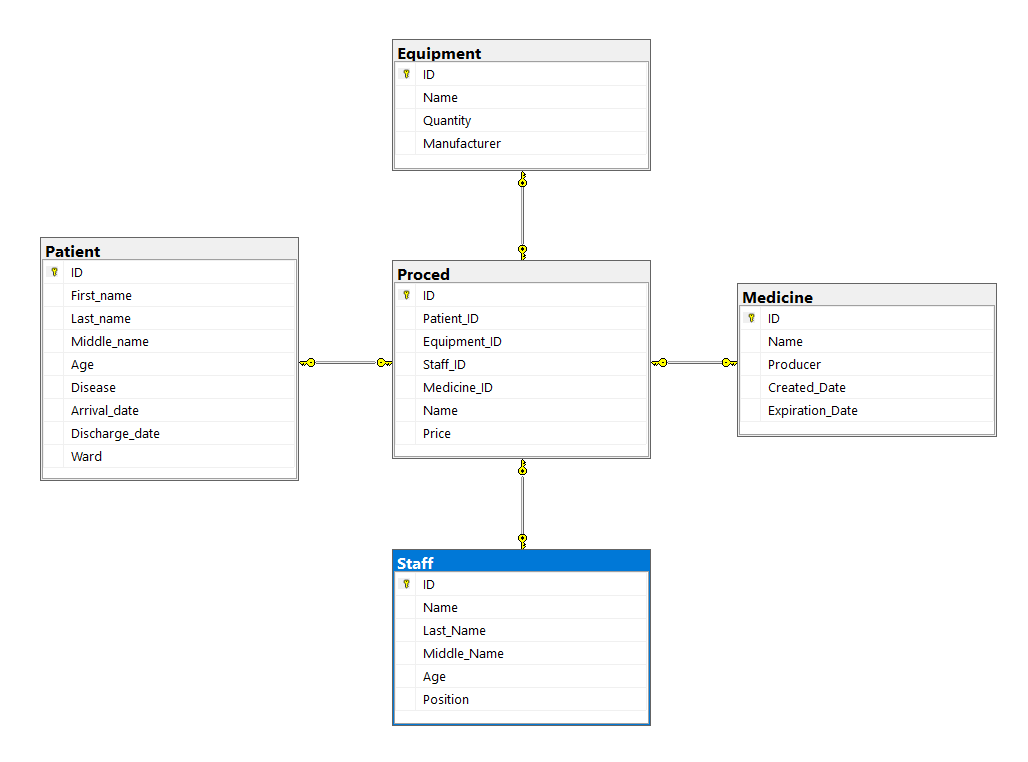
dt.Rows.Add(10, 7, 10, 4, 5, "Ехо серця", 450);

return dt;

}

**Завдання 2:** Об’єднати створені таблиці в об'єкті DataSet реалізувавши зв'язки згідно з індивідуальним завданням.

Зв’язки між таблицями даної бази даних виглядають наступним чином:



В коді, зв’язки реалізовані за допомогою датасетів:

public DataSet GenerateDataSet()

{

DataTable Equip = GetEquipmentTable(); // Тут я читаю всі створені таблиці

DataTable Medicine = GetMedicineTable();

DataTable Patients = GetPatientTable();

DataTable Staff = GetStaffTable();

DataTable Procedures = GetProceduresTable();

DataSet ds = new DataSet("MyDS"); // Створюю датасет і додаю всі таблиці в нього

ds.Tables.Add(Equip);

ds.Tables.Add(Medicine);

ds.Tables.Add(Patients);

ds.Tables.Add(Staff);

ds.Tables.Add(Procedures);

// Виставляю всі зв'язки між таблицями

DataRelation proc\_pat = new DataRelation("ProcPat", ds.Tables["Пацієнти"].Columns["ID"], ds.Tables["Процедури"].Columns["PatientID"]);

DataRelation proc\_equip = new DataRelation("ProcEquip", ds.Tables["Спорядження"].Columns["ID"], ds.Tables["Процедури"].Columns["EquipmentID"]);

DataRelation proc\_staff = new DataRelation("ProcStaff", ds.Tables["Персонал"].Columns["ID"], ds.Tables["Процедури"].Columns["StaffID"]);

DataRelation proc\_med = new DataRelation("ProcMed", ds.Tables["Ліки"].Columns["ID"], ds.Tables["Процедури"].Columns["MedID"]);

// Зберігаю зв'зки в датасеті

ds.Relations.Add(proc\_pat);

ds.Relations.Add(proc\_equip);

ds.Relations.Add(proc\_staff);

ds.Relations.Add(proc\_med);

return ds;

}

**Завдання 3:** Вивести дані з таблиць користувачу

Отже, для реалізації виведення даних була створена фунцкія Display().

Вона виглядає наступним чином:

public static void Display(DataTable dt, DataSet ds)

{

Console.WriteLine("Таблиця {0}:", dt.TableName);

if (dt.TableName != "Процедури" && ds != null) // Вивджу зміст таблиць

{

foreach (DataRow dr in dt.Rows)

{

foreach (var item in dr.ItemArray)

{

// Перевірка на null

if (item != System.DBNull.Value) Console.Write(item + " ");

else Console.Write("--- ");

}

Console.WriteLine();

}

}

if (dt.TableName == "Процедури" && ds != null) // Для виводу таблиці Процедури треба дивитись в інші таблиці

{

// Отримаю всі релейшини

DataRelation proc\_pat = ds.Relations["ProcPat"];

DataRelation proc\_equip = ds.Relations["ProcEquip"];

DataRelation proc\_staff = ds.Relations["ProcStaff"];

DataRelation proc\_med = ds.Relations["ProcMed"];

foreach (DataRow dr in dt.Rows)

{

int patientID = (int)dr["PatientID"];

int equipmentID = (int)dr["EquipmentID"];

int staffID = (int)dr["StaffID"];

int medID = (int)dr["MedID"];

// Отримую дочірні рядки для кожного ID

DataRow[] patients = ds.Tables["Пацієнти"].Select($"ID = {patientID}");

DataRow[] equipment = ds.Tables["Спорядження"].Select($"ID = {equipmentID}");

DataRow[] staff = ds.Tables["Персонал"].Select($"ID = {staffID}");

DataRow[] medicine = ds.Tables["Ліки"].Select($"ID = {medID}");

if (patients.Length > 0) // Виводжу дані

{

Console.WriteLine();

Console.Write("Пацієнт: {0} {1} {2} {3}", patients[0].ItemArray[2],patients[0].ItemArray[1], patients[0].ItemArray[3], patients[0].ItemArray[5]);

Console.WriteLine();

}

if (equipment.Length > 0)

{

Console.Write("Обладнання: {0} {1}", equipment[0].ItemArray[1], equipment[0].ItemArray[3]);

Console.WriteLine();

}

if (staff.Length > 0)

{

Console.Write("Персонал: {0} {1} {2} {3}", staff[0].ItemArray[2], staff[0].ItemArray[1], staff[0].ItemArray[3], staff[0].ItemArray[5]);

Console.WriteLine();

}

if (medicine.Length > 0)

{

Console.Write("Ліки: {0} {1} {2} {3}", medicine[0].ItemArray[1], medicine[0].ItemArray[2], medicine[0].ItemArray[3], medicine[0].ItemArray[4]);

Console.WriteLine();

}

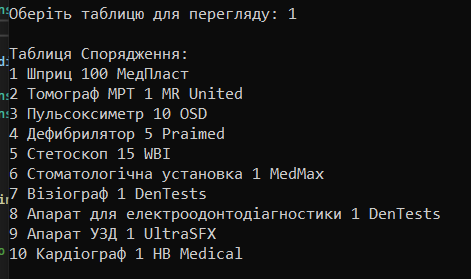
}

}

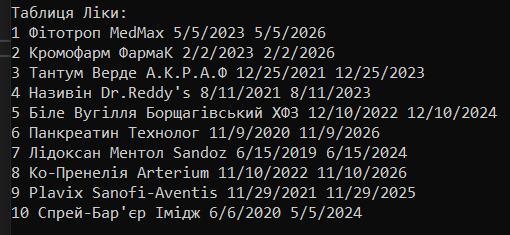
Console.WriteLine();

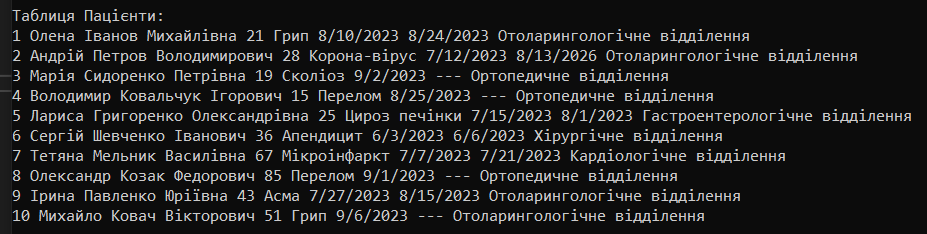
}

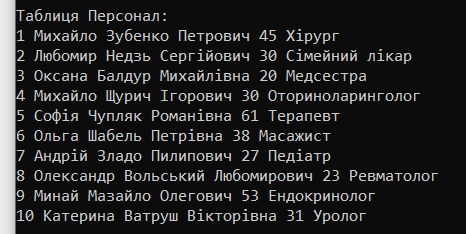
Результат виклику функції буде наступним:

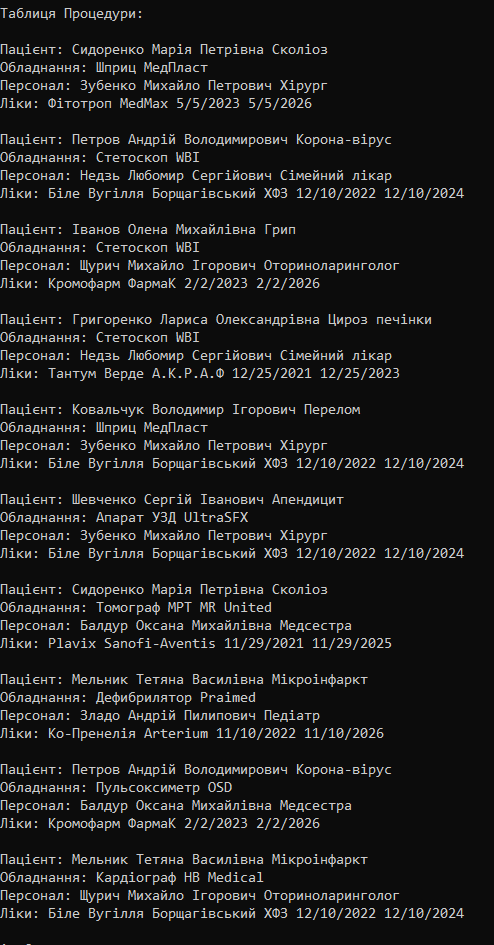


Аналогічно і для всіх інших таблиць:









**Завдання 4:** За допомогою DataView реалізувати сортування та фільтрування однієї з таблиць та вивести дані з цього DataView користувачу.

Для сортування та фільтрування даних було створено окремі функції Filter та Sort. Також було додатковано додано функції меню SortFilterMenu, SortMenu та FilterMenu для зручного користування сортуванням та фільтруванням. Для виведення відсортованої/відфільтрованої інформації, було створено функцію DisplayView.

// Меню сортування/фільтрування

public static void SortFilterMenu(DataSet ds)

{

string choice = "-1";

while (choice != "0")

{

// Вибір методу

Console.WriteLine("\n Сортування/Фільтрування\n\n" +

"1. Сортувати таблицю\n" +

"2. Фільтрувати таблицю\n\n" +

"0. Назад");

Console.Write("\nОберіть пункт методу: ");

choice = Console.ReadLine();

Console.WriteLine();

switch (choice)

{

case "0":

break;

case "1":

SortMenu(ds);

break;

case "2":

FilterMenu(ds);

break;

default:

Console.WriteLine("Неправильний ввід!!!");

break;

}

}

}

// Меню сортування

public static void SortMenu(DataSet ds)

{

int choice;

int tableIndex;

string column;

string sortType;

// Вибір таблиці

for (int i = 0; i < ds.Tables.Count; i++)

{

Console.WriteLine($"{i + 1}. {ds.Tables[i].TableName}");

}

Console.WriteLine("\n0. Назад");

Console.Write("\nОберіть таблицю: ");

choice = Convert.ToInt32(Console.ReadLine());

if (choice < 0 || choice > ds.Tables.Count)

{

Console.WriteLine("Неправильний ввід!!!\n\n");

return;

}

if (choice == 0) return;

tableIndex = choice - 1;

// Вибір стовпця

Console.WriteLine();

for (int i = 0; i < ds.Tables[tableIndex].Columns.Count; i++)

{

Console.WriteLine($"{i + 1}. {ds.Tables[tableIndex].Columns[i].ColumnName}");

}

Console.Write("\nОберіть стовпець: ");

choice = Convert.ToInt32(Console.ReadLine());

if (choice <= 0 || choice > ds.Tables[tableIndex].Columns.Count)

{

Console.WriteLine("Неправильний ввід!!!\n\n");

return;

}

column = ds.Tables[tableIndex].Columns[choice - 1].ColumnName;

// Вибір типу сортування

Console.WriteLine("\n1. За зростанням\n2. За спаданням\n");

Console.Write("Оберіть як сортувати: ");

choice = Convert.ToInt32(Console.ReadLine());

switch (choice)

{

case 0:

return;

case 1:

sortType = "ASC";

break;

case 2:

sortType = "DESC";

break;

default:

Console.WriteLine("Неправильний ввід!!!\n\n");

return;

}

Console.WriteLine();

// Сортування та вивід

Sort(ds.Tables[tableIndex], column, sortType);

}

// Меню фільтрування

public static void FilterMenu(DataSet ds)

{

int choice;

int tableIndex;

string column;

string filterType;

string filterValue;

// Вибір таблиці

for (int i = 0; i < ds.Tables.Count; i++)

{

Console.WriteLine($"{i + 1}. {ds.Tables[i].TableName}");

}

Console.WriteLine("\n0. Назад");

Console.Write("\nОберіть таблицю: ");

choice = Convert.ToInt32(Console.ReadLine());

if (choice < 0 || choice > ds.Tables.Count)

{

Console.WriteLine("Неправильний ввід!!!\n\n");

return;

}

if (choice == 0) return;

tableIndex = choice - 1;

// Вибір стовпця

Console.WriteLine();

for (int i = 0; i < ds.Tables[tableIndex].Columns.Count; i++)

{

Console.WriteLine($"{i + 1}. {ds.Tables[tableIndex].Columns[i].ColumnName}");

}

Console.Write("\nОберіть стовпець: ");

choice = Convert.ToInt32(Console.ReadLine());

if (choice <= 0 || choice > ds.Tables[tableIndex].Columns.Count)

{

Console.WriteLine("Неправильний ввід!!!\n\n");

return;

}

column = ds.Tables[tableIndex].Columns[choice - 1].ColumnName;

// Вибір методу фільтрування

Console.WriteLine("\n1. За вмістом у стрічці стовпця\n2. Більше, ніж...\n3. Менше, ніж...");

Console.Write("\nОберіть як фільтрувати: ");

choice = Convert.ToInt32(Console.ReadLine());

switch (choice)

{

case 0:

return;

case 1:

filterType = "LIKE";

break;

case 2:

filterType = ">";

break;

case 3:

filterType = "<";

break;

default:

Console.WriteLine("Неправильний ввід!!!\n\n");

return;

}

// Вводимо знчення фільтрації

Console.Write("\nВведіть значення для фільтрування: ");

filterValue = Console.ReadLine();

if (filterType == "LIKE") filterValue = $"\'%{filterValue}%\'";

Console.WriteLine();

// Фільтрація та вивід

Filter(ds.Tables[tableIndex], column, filterType, filterValue);

}

// Сортування, передаємо таблицю, стовпець для сортування та як сортувати (спадання/зростання)

public static void Sort(DataTable dt, string sortColumn, string sortMethod)

{

DataView dv = new DataView(dt);

dv.Sort = $"{sortColumn} {sortMethod}";

DisplayView(dv);

}

// Фільтрування, передаємо таблицю, стовпець для фільтрування, як фільтрувати, та значення для фільтрування

public static void Filter(DataTable dt, string filterColumn, string filterMethod, string filterValue)

{

DataView dv = new DataView(dt);

try

{

dv.RowFilter = $"[{filterColumn}] {filterMethod} {filterValue}";

}

// Якщо є помилка

catch (Exception)

{

Console.WriteLine("Не знайшлось рядків, які підлягають критеріям :(\n");

return;

}

// Якщо не знайшло рядків

if (dv.ToTable().Rows.Count == 0)

{

Console.WriteLine("Не знайшлось рядків, які підлягають критеріям :(\n");

return;

}

// Виведення таблиці

DisplayView(dv);

}

// Виведення таблиці

public static void DisplayView(DataView dv)

{

foreach (DataRowView dr in dv)

{

foreach (DataColumn dc in dv.Table.Columns)

{

if (dr.Row[dc] != System.DBNull.Value) Console.Write(dr.Row[dc] + " ");

else Console.Write("--- ");

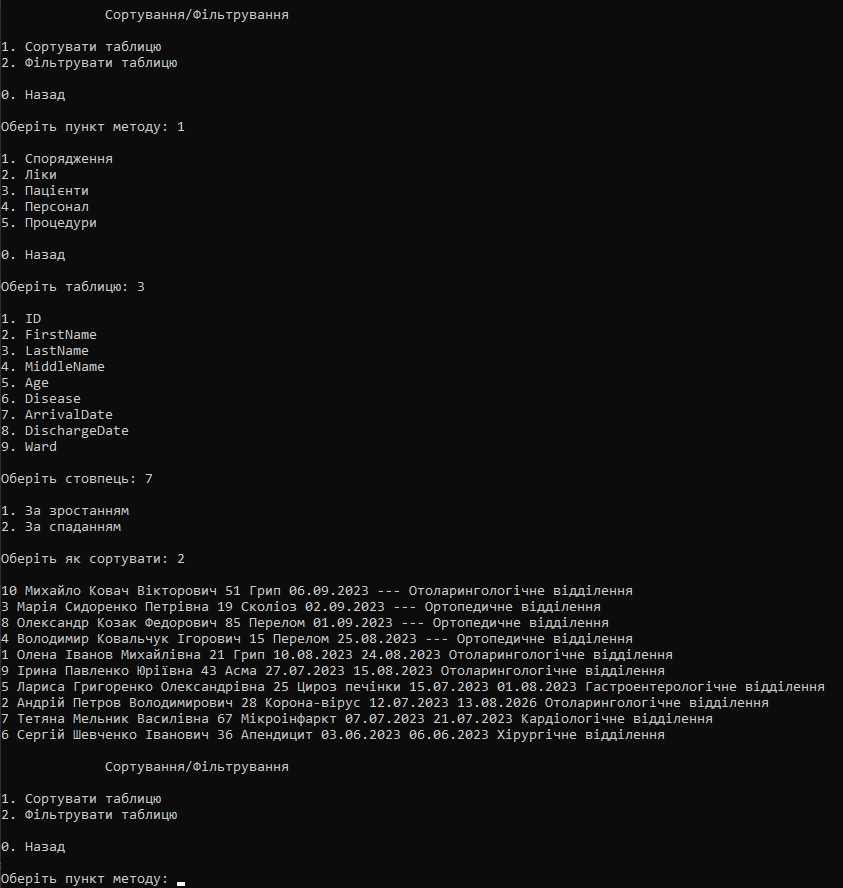
}

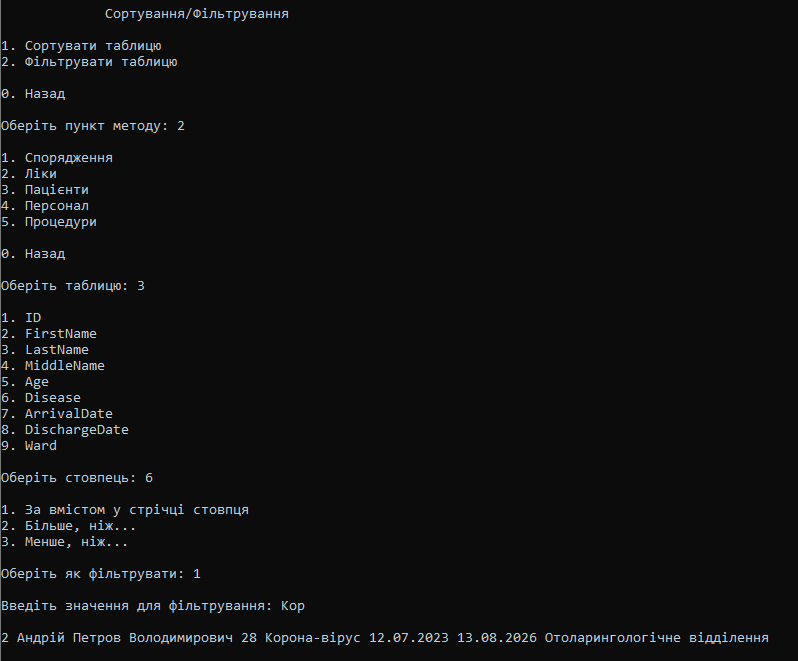
Console.WriteLine();

}

}

Результат виконання:





**Завдання 5:** Внести зміни в кожну з таблиць(додати, видалити і модифікувати, як мінімум один рядок).

Для реалізації даного функціоналу було створено функції AddRow, DeleteRow та ModifyRow відповідно. Для зручного користування також було створено функції меню ChooseTableToEditMenu та TableEditMenu

// Меню вибору таблиці для редагування

public static void ChooseTableToEditMenu(DataSet ds)

{

int choice = -1;

while (choice != 0)

{

// Вибір таблиць

for (int i = 0; i < ds.Tables.Count; i++)

{

Console.WriteLine($"{i + 1}. {ds.Tables[i].TableName}");

}

Console.WriteLine("\n0. Назад");

Console.Write("\nОберіть таблицю для редагування: ");

choice = Convert.ToInt32(Console.ReadLine());

// Вибір таблиці. В функцію передаються типи даних таблиці, та підказки для вводу. НЕ МІНЯТИ

Console.WriteLine();

switch (choice)

{

case 0:

return;

case 1:

TableEditMenu(ds.Tables[choice - 1],

new string[] { "string", "int", "string" },

new string[] { "Назва: ", "Кількість: ", "Виробник:" });

break;

case 2:

TableEditMenu(ds.Tables[choice - 1],

new string[] { "string", "string", "date", "date" },

new string[] { "Назва: ", "Виробник: ", "Дата виробництва: ", "Дата 'вжити до': " });

break;

case 3:

TableEditMenu(ds.Tables[choice - 1],

new string[] { "string", "string", "string", "int", "string", "date", "date", "string" },

new string[] { "Ім'я: ", "Прізвище: ", "По-батькові: ", "Вік: ", "Хвороба: ", "Дата прибуття: ", "Дата виписки: ", "Відділення: " });

break;

case 4:

TableEditMenu(ds.Tables[choice - 1],

new string[] { "string", "string", "string", "int", "string" },

new string[] { "Ім'я: ", "Прізвище: ", "По-батькові: ", "Вік: ", "Посада: " });

break;

case 5:

TableEditMenu(ds.Tables[choice - 1],

new string[] { "int", "int", "int", "int", "string", "int" },

new string[] { "Номер пацієнта: ", "Номер спорядження: ", "Номер лікаря: ", "Номер ліків: ", "Назва: ", "Ціна: " });

break;

default:

Console.WriteLine("Неправильний ввід!!!\n\n");

break;

}

}

}

// Меню вибору типу редагування таблиці

public static void TableEditMenu(DataTable dt, string[] dataTypes, string[] prompts)

{

int choice = -1;

while (choice != 0)

{

// Вибір типу редагування

Console.WriteLine("1. Додати новий рядок\n" +

"2. Редагувати рядок таблиці\n" +

"3. Видалити рядок\n\n" +

"0. Назад");

Console.Write("\nОберіть пункт редагування таблиці: ");

choice = Convert.ToInt32(Console.ReadLine());

Console.WriteLine();

switch (choice)

{

case 0:

return;

case 1:

AddRow(dt, dataTypes, prompts);

break;

case 2:

ModifyRow(dt, dataTypes, prompts);

break;

case 3:

DeleteRow(dt);

break;

default:

Console.WriteLine("Неправильний ввід!!!\n\n");

break;

}

}

}

// Додавання нового рядку до таблиці, передаємо таблицю, типи стовпців, та підказки для вводу

public static void AddRow(DataTable dt, string[] dataTypes, string[] prompts)

{

// Новий рядок

DataRow row = dt.NewRow();

if (dt.Rows.Count != 0) row[0] = Convert.ToInt32(dt.Rows[dt.Rows.Count - 1].ItemArray[0]) + 1;

else row[0] = 1;

// Заповнення рядка

for (int i = 0; i < dataTypes.Length; i++)

{

// Підказка для вводу

Console.Write(prompts[i]);

// Заповнення стовпця в залежності від типу

switch (dataTypes[i])

{

case "int":

row[i + 1] = Convert.ToInt32(Console.ReadLine());

break;

case "string":

row[i + 1] = Console.ReadLine();

break;

case "date":

string[] temp = Console.ReadLine().Split(".");

row[i + 1] = new DateOnly(Convert.ToInt32(temp[2]), Convert.ToInt32(temp[1]), Convert.ToInt32(temp[0]));

break;

default:

row[i + 1] = Console.ReadLine();

break;

}

}

// Занесення рядка до таблиці

dt.Rows.Add(row);

WriteChanges(row, changesDS.Tables["Створені"], dt.TableName);

Console.WriteLine();

}

// Зміна рядка в таблиці

public static void ModifyRow(DataTable dt, string[] dataTypes, string[] prompts)

{

// Виводимо таблицю

DataRow row;

DataView dv = new DataView(dt);

DisplayView(dv);

Console.WriteLine("\n0. Назад");

// Вказуємо рядок

int choice;

Console.Write("\nВведіть рядок, який потрібно змінити: ");

choice = Convert.ToInt32(Console.ReadLine());

if (choice < 0 || choice > dt.Rows.Count)

{

Console.WriteLine("Неправильний ввід!!!\n\n");

return;

}

if (choice == 0)

{

return;

}

// Призначаємо рядок для змін

row = dt.Rows[choice - 1];

Console.WriteLine("Якщо бажаєте залишити стовпець без змін, залиште ввід пустим~\n");

for (int i = 0; i < dataTypes.Length; i++)

{

// Приймаємо значення з консолі

Console.Write(prompts[i]);

string input = Console.ReadLine();

if (input == "") continue;

// Заміняємо значення стовпця у рядку

switch (dataTypes[i])

{

case "int":

row[i + 1] = Convert.ToInt32(input);

break;

case "string":

row[i + 1] = input;

break;

case "date":

string[] temp = input.Split(".");

row[i + 1] = new DateOnly(Convert.ToInt32(temp[2]), Convert.ToInt32(temp[1]), Convert.ToInt32(temp[0]));

break;

default:

row[i + 1] = input;

break;

}

}

WriteChanges(row, changesDS.Tables["Модифіковані"], dt.TableName);

Console.WriteLine();

}

// Видалення рядка з таблиці

public static void DeleteRow(DataTable dt)

{

// Виведення таблиці

DataView dv = new DataView(dt);

DisplayView(dv);

Console.WriteLine("\n0. Назад");

// Вибір рядка

int choice;

Console.Write("\nВведіть рядок, який потрібно видалити: ");

choice = Convert.ToInt32(Console.ReadLine());

if (choice < 0 || choice > dt.Rows.Count)

{

Console.WriteLine("Неправильний ввід!!!\n\n");

return;

}

if (choice == 0)

{

return;

}

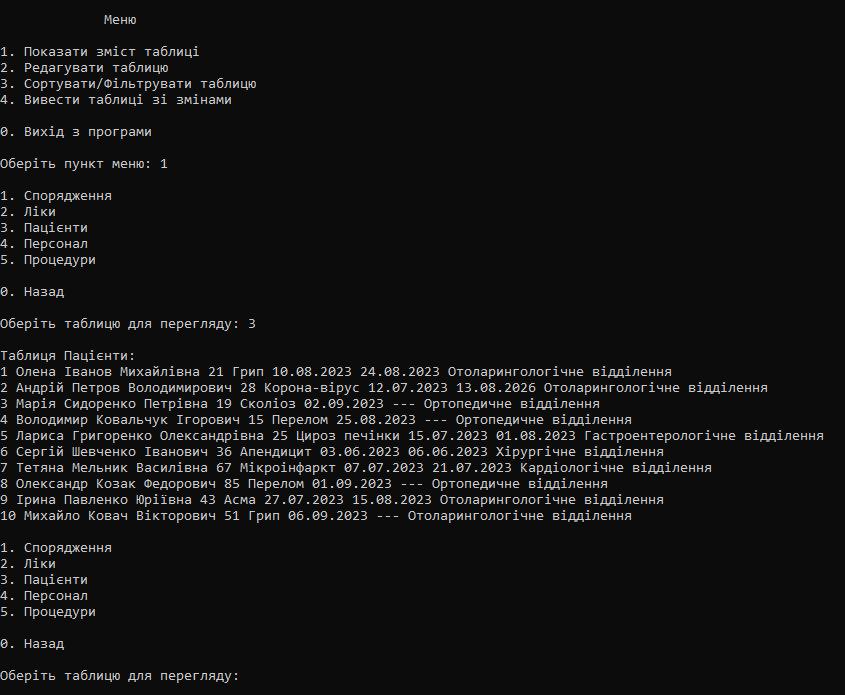
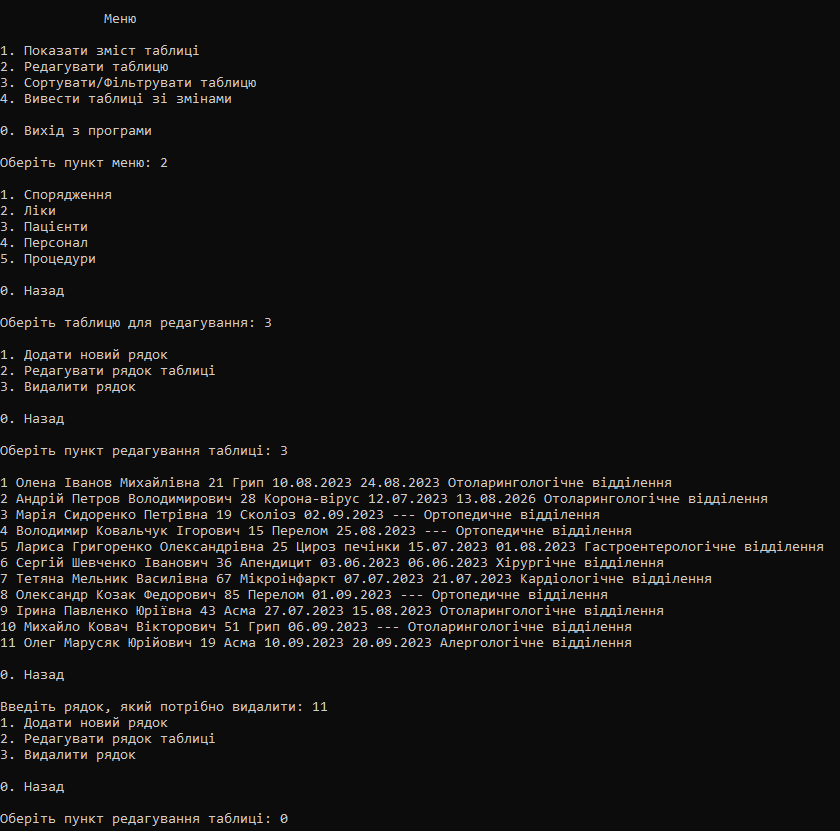
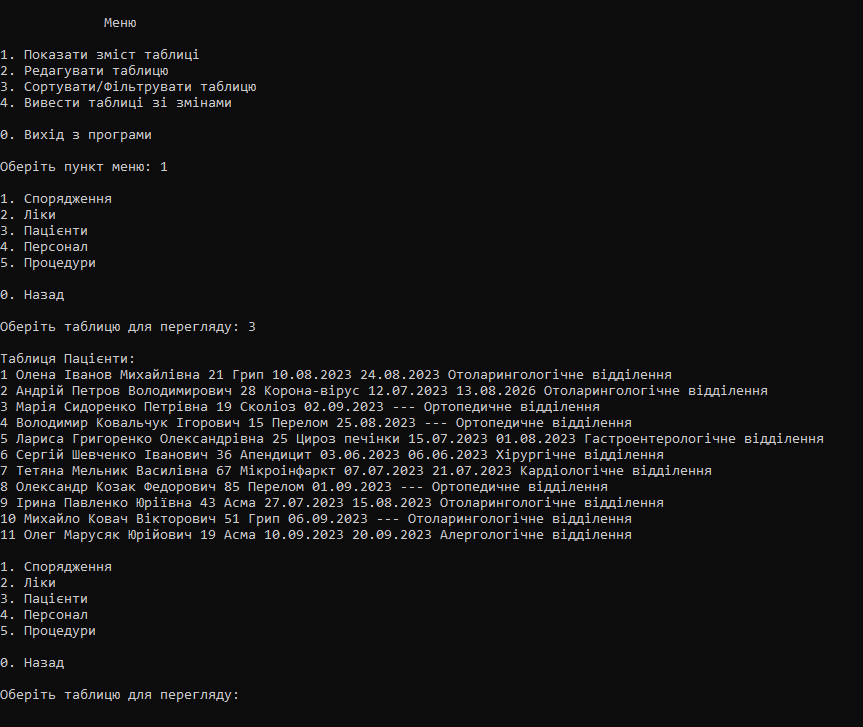
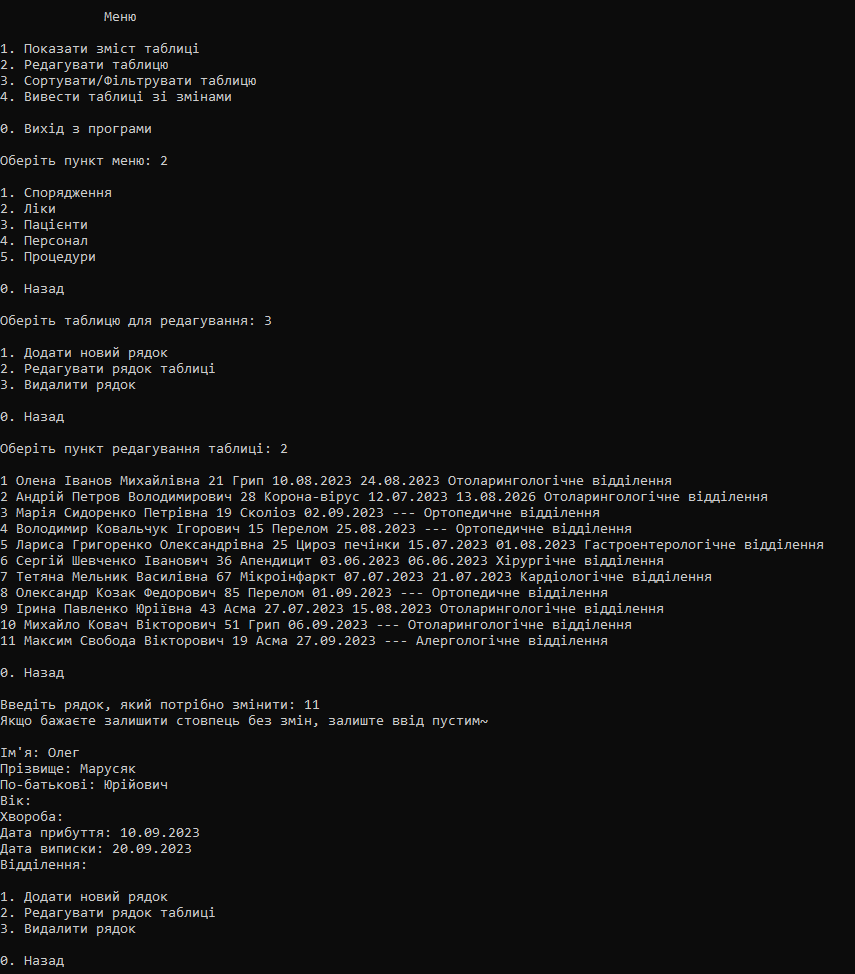
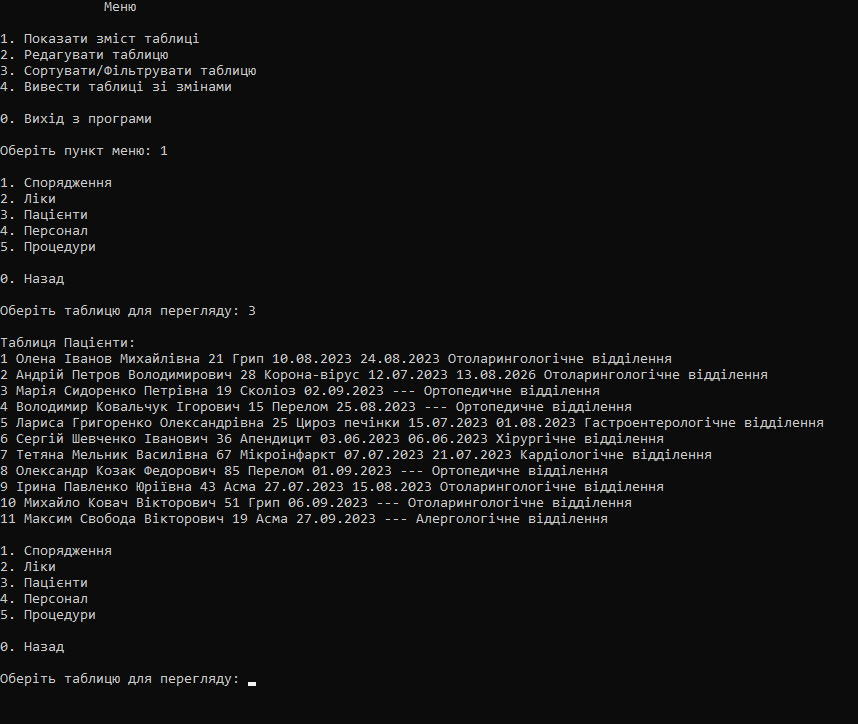
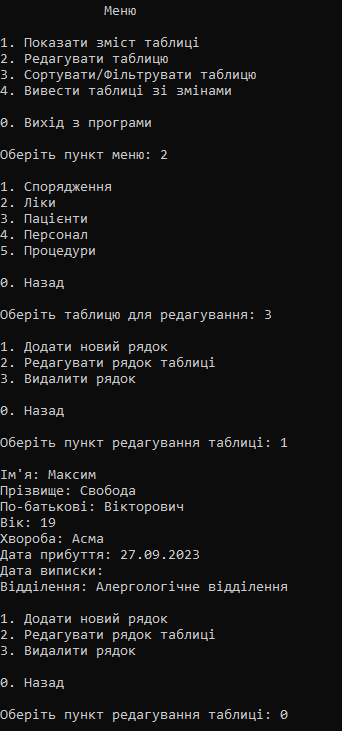
WriteChanges(dt.Rows[choice - 1], changesDS.Tables["Видалені"], dt.TableName);

// Видалення рядка за індексом

dt.Rows.RemoveAt(choice - 1);

}

Результат виконання:



**Завдання 6:** Сформувати новий DataSet, який міститиме тільки змінені дані(додані, модифіковані, видалені) та відобразити його користувачу при цьому відобразити інформацію тип змін, що відбулись в кожному з рядків(додавання/модифікація/видалення).

Для реалізації вимог було створено функцію GenerateChangesDataSet, що створює новий DataSet із трьома таблицями, а також функцію WriteChanges, яка викликається в той момент, коли здійснюється зміна в будь-якій з таблиць.

public DataSet GenerateChangesDataSet()

{

DataTable dt1 = new DataTable("Створені");

DataColumn dc1 = new DataColumn("ID", typeof(int));

dt1.Columns.Add(dc1);

dt1.PrimaryKey = new DataColumn[] { dc1 };

dt1.Columns.Add(new DataColumn("Table", typeof(string)));

dt1.Columns.Add(new DataColumn("C1", typeof(string)));

dt1.Columns.Add(new DataColumn("C2", typeof(string)));

dt1.Columns.Add(new DataColumn("C3", typeof(string)));

dt1.Columns.Add(new DataColumn("C4", typeof(string)));

dt1.Columns.Add(new DataColumn("C5", typeof(string)));

dt1.Columns.Add(new DataColumn("C6", typeof(string)));

dt1.Columns.Add(new DataColumn("C7", typeof(string)));

dt1.Columns.Add(new DataColumn("C8", typeof(string)));

DataTable dt2 = new DataTable("Модифіковані");

DataColumn dc2 = new DataColumn("ID", typeof(int));

dt2.Columns.Add(dc2);

dt2.PrimaryKey = new DataColumn[] { dc2 };

dt2.Columns.Add(new DataColumn("Table", typeof(string)));

dt2.Columns.Add(new DataColumn("C1", typeof(string)));

dt2.Columns.Add(new DataColumn("C2", typeof(string)));

dt2.Columns.Add(new DataColumn("C3", typeof(string)));

dt2.Columns.Add(new DataColumn("C4", typeof(string)));

dt2.Columns.Add(new DataColumn("C5", typeof(string)));

dt2.Columns.Add(new DataColumn("C6", typeof(string)));

dt2.Columns.Add(new DataColumn("C7", typeof(string)));

dt2.Columns.Add(new DataColumn("C8", typeof(string)));

DataTable dt3 = new DataTable("Видалені");

DataColumn dc3 = new DataColumn("ID", typeof(int));

dt3.Columns.Add(dc3);

dt3.PrimaryKey = new DataColumn[] { dc3 };

dt3.Columns.Add(new DataColumn("Table", typeof(string)));

dt3.Columns.Add(new DataColumn("C1", typeof(string)));

dt3.Columns.Add(new DataColumn("C2", typeof(string)));

dt3.Columns.Add(new DataColumn("C3", typeof(string)));

dt3.Columns.Add(new DataColumn("C4", typeof(string)));

dt3.Columns.Add(new DataColumn("C5", typeof(string)));

dt3.Columns.Add(new DataColumn("C6", typeof(string)));

dt3.Columns.Add(new DataColumn("C7", typeof(string)));

dt3.Columns.Add(new DataColumn("C8", typeof(string)));

DataSet ds = new DataSet("ChangesDS");

ds.Tables.Add(dt1);

ds.Tables.Add(dt2);

ds.Tables.Add(dt3);

return ds;

}

// Запис змін в інший датасет

public static void WriteChanges(DataRow row, DataTable changesTable, string tableName)

{

// Новий рядок для таблиці

DataRow newChangeRow = changesTable.NewRow();

// Запис ID

if (changesTable.Rows.Count != 0) newChangeRow[0] =

Convert.ToInt32(changesTable.Rows[changesTable.Rows.Count - 1].ItemArray[0]) + 1;

else newChangeRow[0] = 1;

// Запис таблиці

newChangeRow[1] = tableName;

int i = 1;

// Додавання інформації рядка

for (; i < row.ItemArray.Length; i++)

{

newChangeRow[i + 1] = row[i];

}

// Дозаповнення пустих рядків

for (; i < changesTable.Columns.Count - 1; i++)

{

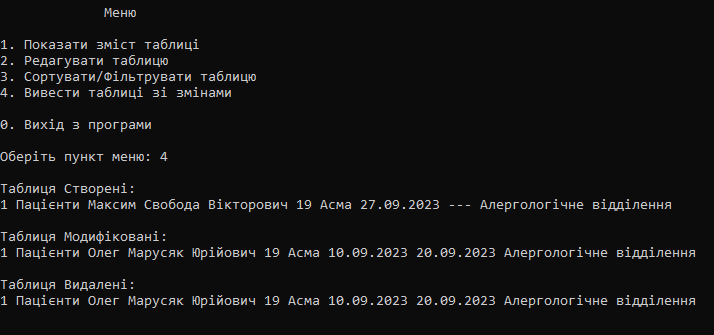
newChangeRow[i + 1] = null;

}

changesTable.Rows.Add(newChangeRow);

}

Результат виконання:



**Висновок:** На цій лабораторній роботі ми ознайомились зі структурою класів ADO.NET, здобули практичні навички роботи з класами ADO.NET, що можуть функціонувати без підключення до БД, зокрема DataTable, DataColumn, DataRow, DataSet та інші.