МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

«ПОЛТАВСЬКА ПОЛІТЕХНІКА ІМЕНІ ЮРІЯ КОНДРАТЮКА»

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА РОБОТОТЕХНІКИ

КАФЕДРА КОМП’ЮТЕРНИХ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І СИСТЕМ

**Розрахунково-графічна робота (РГР)**

**з дисципліни**

**«Об’єктно-орієнтоване програмування»**

**спеціальність 123 «Комп’ютерна інженерія »**

**студента групи 202-ТК Базавлук Максим Віталійович**

Керівник роботи

старший викладач

кафедри комп’ютерних

та інформаційних

технологій і систем

Демиденко М.І.

**м. Полтава 2024**

**ЗМІСТ**

**Вступ 3**

**Розділ 1 ПОСТАНОВКА ЗАДАЧІ 6**

**1.2 Мова програмування C# 6**

* 1. **Поняття про «інфекція стригучого лишая» 8**
  2. **Реалізація моделювання поширення інфекції через мову C# 9**

**Розділ 2 ОПИС ПРОЕКТУ 10**

**Розділ 3 МОДЕЛЮВАННЯ ПРОЦЕСУ ПОШИРЕННЯ ІНФЕКЦІЇ 13**

**3.1 Основне вікно моделювання заражених клітин 13**

**3.2 Основне вікно моделювання несприятливих клітин(імунні) 14**

**3.3 Основне вікно моделювання здорових клітин 15**

**3.4 Основний код нашої програми 16**

**ВИСНОВКИ 21**

**СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 22**

**ВСТУП**

ООП (об'єктно-орієнтоване програмування) є важливим поняттям у світі програмування. Воно є підхідом до розробки програмного забезпечення, який дозволяє створювати програми, орієнтовані на об'єкти, які взаємодіють між собою.

ООП ґрунтується на чотирьох основних принципах: інкапсуляція, успадкування, поліморфізм та абстракція. Давай детальніше розглянемо кожен із них.

* Інкапсуляція. Дані та методи, пов’язані з ними, зберігаються всередині об’єкта.
* Успадкування. В ООП можна створювати нові класи на основі вже наявних.
* Поліморфізм. Об’єкти різних класів можуть мати однакові методи, але реалізовувати їх різними способами.
* Абстракція. Замість того щоб детально описувати кожну частину системи, абстракція фокусується на найважливішому.

Об'єктно-орієнтоване програмування сягає своїм корінням до створення мови програмування Симула в 1960-х роках, одночасно з посиленням дискусій про кризу програмного забезпечення. Через ускладнення апаратного та програмного забезпечення було дуже важко зберегти якість програм. Об'єкто-орієнтоване програмування частково розв'язує цю проблему шляхом наголошення на модульності програми.

В основі об’єктно-орієнтованої мови програмування лежать два основних поняття: клас та об’єкт. Об’єкт – це базове поняття в ООП, це конкретна реалізація, екземпляр класу. Об’єкт складається з трьох частин: стан (змінні стану), методи (операції), ім’я об’єкта. Клас – це група даних і методів або функцій для роботи з цими даними, це шаблон. Об’єкти однаковими наборами змінних стану і методів, утворюють клас. Якщо об’єкти мають реалізацію з конкретного світу, то класи є абстракціями. Трохи більш складні об’єкти можуть взагалі не містити даних, а представляти процес і містити тільки функції, які реалізують цей процес. Для формування реального об’єкта необхідно мати шаблон, по прикладу якого і будується даний об’єкт.

Принципи об’єктно-орієнтованого програмування широко застосовуються в різних галузях розробки. Ось декілька прикладів із реального життя:

* [Веброзробка.](https://goit.global/ua/courses/fullstack/) Класи, об’єкти, успадкування і поліморфізм дають змогу заводити різні типи користувачів, товарів, замовлень та інших сутностей на основі загальних шаблонів. Це корисно, наприклад, при створенні соціальних мереж або інтернет-магазинів.
* Розробка ігор. Щоб додавати персонажів, а також предмети, світи та інші елементи із загальними характеристиками.
* Медицина. ООП застосовується для розробки медичних інформаційних систем, які зберігають і опрацьовують дані. Класи та об’єкти представляють пацієнтів, лікарів, ліки тощо. Інкапсуляція забезпечує безпечне зберігання та доступ до них.

Об'єктно-орієнтоване програмування (ООП) підтримується багатьма мовами програмування. Ось деякі з них:

1. **C++**: Одна з найпопулярніших мов, що підтримує ООП, використовується для системного програмування, розробки ігор, додатків та інших програмних продуктів.
2. **Java**: Повністю об'єктно-орієнтована мова, широко використовується для розробки веб-додатків, корпоративних додатків, мобільних додатків на платформі Android.
3. **C#**: Мова від Microsoft, яка також є повністю об'єктно-орієнтованою. Використовується для розробки на платформі .NET, веб-додатків, ігор за допомогою Unity.
4. **Python**: Підтримує ООП разом з іншими парадигмами програмування, широко використовується в науці про дані, веб-розробці, автоматизації та багатьох інших областях.
5. **Ruby**: Мова, яка підтримує ООП і відома своєю простотою та потужністю. Використовується для розробки веб-додатків, зокрема, з використанням фреймворку Ruby on Rails.
6. **Swift**: Мова програмування від Apple, яка підтримує ООП, використовується для розробки додатків під iOS та macOS.
7. **PHP**: Хоча не завжди асоціюється з ООП, сучасні версії PHP підтримують об'єктно-орієнтовані концепції. Використовується для веб-розробки.
8. **TypeScript**: Надмножина JavaScript, яка додає статичну типізацію і повну підтримку ООП. Широко використовується у великих проектах, де необхідний більш суворий контроль типів.

Це лише кілька прикладів, і існує багато інших мов, які також підтримують об'єктно-орієнтоване програмування.

**РОЗДІЛ 1**

**ПОСТАНОВКА ЗАДАЧІ**

Завдання про інфекцію стригучого лишаю (Ван Тассел Д. Стиль, розробка, ефективність, налагодження й випробування програм. - М.: Мир, 1981). Промоделювати процес поширення інфекції — стригучого лишаю по ділянці шкіри розміром n х n (n — непарне) клітинок. Передбачається, що вихідною зараженою клітинкою шкіри є центральна. У кожний інтервал часу уражена інфекцією клітинка може з імовірністю 0,5 заразити кожну із сусідніх здорових клітинок. По закінченні шести одиниць часу заражена клітинка стає несприйнятливою до інфекції імунітет, що виник, діє протягом наступних чотирьох одиниць часу, а потім клітинка виявляється здоровою. У ході моделювання описаного процесу видавати поточний стан ділянки шкіри, що моделюється, в кожному інтервалі часу, відзначаючи заражені, несприйнятливі до інфекції й здорові клітинки.

* 1. **Мова програмування C#**



Рис.1.1 Мова C#

Мова програмування C# - це об'єктноорієнтована й багатопарадигмальна мова для створення сайтів, програм та ігор.

C# вимовляється «Сі-шарп». Назву взяли з музичної нотації, де символ «#» – октоторп або дієз – вказує на те, що ноту слід зіграти на півтону вище. Суфікс «шарп» також використовувався кількома іншими мовами програмування платформи .NET, а саме виданнями сучасних мов, наприклад, J#, A# та функціональна мова програмування F#.

Базовий синтаксис C# подібний до мов стилю C, таких як C, C++ і Java. Ця мова програмування найбільше відповідає стандарту Common Language Infrastructure (CLI).

Інтеграція з .NET також означає доступ до великої бібліотеки класів, яка містить готові рішення для безлічі завдань. Це скорочує час розробки і дозволяє створювати більш функціональні додатки.

* Ця мова програмування підтримує безліч різних платформ, включно з Windows, macOS і Linux. Це означає, що ви можете створювати додатки, які працюють практично скрізь.
* Завдяки інструментам, таким як .NET Core і .NET 5 (та їхнім наступним версіям), C# розробники можуть створювати крос-платформні додатки, які не залежать від конкретної операційної системи. Це особливо важливо в сучасному світі, де різноманітність пристроїв і платформ перебуває на піку.

C# має тісну інтеграцію з операційною системою Windows, що робить його чудовим вибором для створення настільних додатків під цю платформу. За допомогою технології Windows Forms або більш сучасної Universal Windows Platform (UWP), ви можете розробляти додатки з графічним інтерфейсом користувача (GUI), які інтегровані з операційною системою Windows.

**1.2** **Поняття про «інфекція стригучого лишая»**

Стригучий лишай – це грибкове захворювання шкіри, яке належить до групи хвороб під загальною назвою «дерматомікози». Ця патологія виникає внаслідок ураження шкіри нитчастими грибами роду трихофітон (трихофітія) або мікроспорум (мікроспорія).

Цим захворюванням можна заразитися в будь-якому віці під час контакту з хворою людиною або твариною. Варто зазначити, що маленькі діти та вагітні жінки перебувають у групі ризику через фізіологічні особливості імунної системи.

Стригучий лишай відноситься до типу заразних захворювань шкіри. Носіями збудників можуть виступати як люди, так і тварини. Це грибкове інфекційне захворювання, яке вражає верхні шари епідермісу.

Інфекція передається контактним шляхом, у найбільшій групі ризику знаходяться маленькі діти та вагітні, проте стригучий лишай може вражати шкіру дорослої людини, особливо при зниженому імунітеті або наявності травмованої шкіри.



Рис. 1.2 Інфекція лишаю

Процес лікування стригучого лишаю тривалий, зазвичай він триває щонайменше шість тижнів, при цьому системна терапія триває переважно 15-25 днів.

**1.3 Реалізація моделювання поширення інфекції через мову C#**

Для реалізації моделювання поширення інфекції на мові програмування C# з використанням об'єктно-орієнтованого підходу, ми створимо форму для моделювання інфекції. Ця програма буде симулювати поширення інфекції по ділянці шкіри розміром n×n клітинок з центру, як описано у завданні.

Кроки реалізації:

1. Клас Cell:

* Має три стани: здоровий, заражений, імунний.
* Методи Infect і Update для зміни стану клітинки та оновлення її стану відповідно до правил.

1. Клас Grid:

* Зберігає двовимірний масив клітинок.
* Метод RunSimulation для запуску симуляції.
* Методи UpdateGrid і InfectNeighbors для оновлення станів клітинок і зараження сусідніх клітинок.
* Метод PrintGrid для відображення поточного стану сітки.

1. Головний клас Program:

* Створює об'єкт Grid і запускає симуляцію.

**РОЗДІЛ 2**

**ОПИС ПРОЕКТУ**

Дана розрахунково-графічна робота (РГР) включає в себе проекти в середовищі розробки Visual Studio 2022: проект Windows Forms (.NET Framework 4.8) під назвою “РГР(Базавлук)”.

Ми запускаємо Visual Studio 2022 і створюємо новий проект Windows Forms ( з .NET Framework) розширенням і даємо назву нашому проекту.

Наш проект буде мати два основних файла:

1. Файл Form1.cs – форма основного вікна програми, клас якої описує події натискання бокових вкладок (кнопок). Допоміжний клас FormDesignSchema.cs використовується для опису дизайну інтерфейсу користувача цього вікна.
2. Клас Program.cs – точка входу програми.

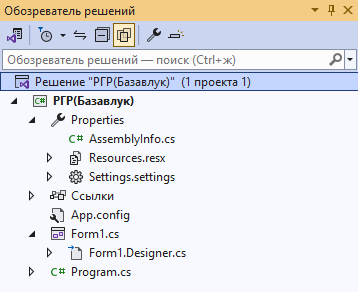


Рис.2.1 Структура проекту WindowsForms

Для створення моделювання в нашому проекті будуть застосовані класи такі як “Cell” та “Grid”.

У класі “Cell” представлені клітинки на ділянці шкіри. Кожен об'єкт класу “Cell” відповідає одній конкретній клітинці і містить інформацію про її поточний стан та час інфекції.

Основні елементи класу Cell включають:

1. Властивість State: Це перечислення, яке визначає стани клітинки (здоровий, заражений, імунний).
2. Властивість InfectionTime: Ціле число, що вказує час, протягом якого клітина перебуває у стані інфікованості або імунності.
3. Метод Infect(): Викликається для зараження клітинки. Під час виклику цього методу стан клітинки змінюється на "заражений", а час інфекції встановлюється на 1.
4. Метод Update(): Оновлює стан клітинки з плином часу. Він збільшує час інфекції та встановлює стан клітинки відповідно до правил моделювання.

У класі “Cell” імплементується логіка, що відбувається з кожною клітинкою під час моделювання поширення інфекції. Цей клас відповідає за індивідуальні клітинки на ділянці шкіри і дозволяє відслідковувати їхній стан та час інфекції.

Основна задача класу “Grid” полягає в моделюванні поширення інфекції серед цих клітинок та оновленні їх станів з плином часу.

В класі “Grid” зазвичай реалізовані наступні функціональності:

1. Створення та ініціалізація масиву клітинок: Клас Grid може містити двовимірний масив об'єктів класу Cell, де кожен елемент представляє одну клітинку на ділянці шкіри. Цей масив створюється та ініціалізується у конструкторі класу.
2. Запуск симуляції: Метод RunSimulation може бути відповідальним за запуск симуляції поширення інфекції на ділянці шкіри. Цей метод може викликати інші методи класу Grid для оновлення станів клітинок та відображення поточного стану сітки.
3. Оновлення стану клітинок: Метод UpdateGrid може відповідати за оновлення стану кожної клітинки відповідно до правил моделювання інфекції. Цей метод може також включати логіку зараження сусідніх клітинок, якщо одна з них заражена.
4. Відображення поточного стану сітки: Метод PrintGrid може бути використаний для відображення поточного стану ділянки шкіри, показуючи, які клітинки здорові, заражені або імунні.

Крім цього, клас “Grid” може містити інші методи та властивості, які допомагають управляти симуляцією, налаштовувати параметри моделювання, зберігати дані про стан клітинок та інше. В загальному, клас “Grid” виконує ключову роль у моделюванні поширення інфекції та управлінні станом ділянки шкіри під час симуляції.

**РОЗДІЛ 3**

**МОДЕЛЮВАННЯ ПРОЦЕСУ ПОШИРЕННЯ ІНФЕКЦІЇ**

* 1. **Основне вікно моделювання заражених клітин**

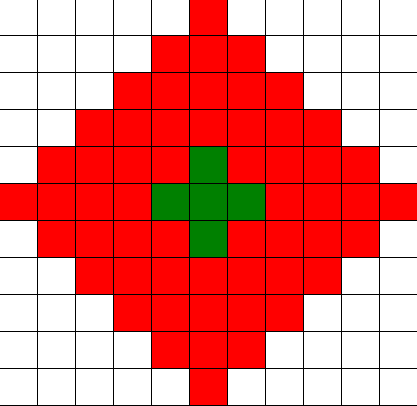
****

Рис.3.1 Заражені клітини

Такий вигляд матиму заражені клітини вони виглядають червоним кольором , такій людині потрібне негайне лікування інфекції лишаю.

* 1. **Основне вікно моделювання несприятливих клітин(імунні)**

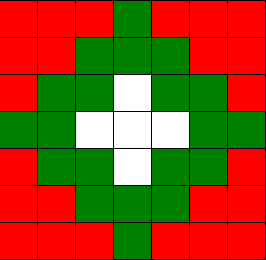


Рис.3.2 Несприятливі клітини (імунні)

Зеленим кольором виглядаю клітини які несприятливі для організму людини , в такому випадку потрібно негайно звертатися до лікаря.

* 1. **Основне вікно моделювання здорових клітин**

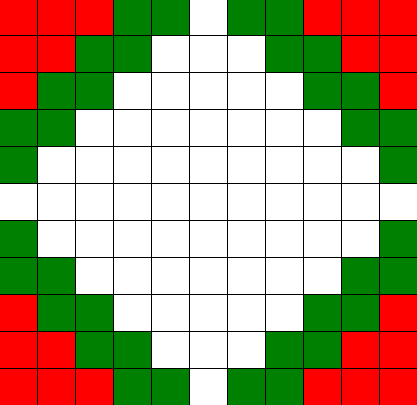
****

Рис.3.3 Здорові клітини

Білим кольором виглядають клітини здорової людини. Інфекція не загрожує , але організм людини як ми можемо бачити по вигляду нашого основного моделювання , знаходиться в нестабільному розвитку. І заразитися інфекцією лишаю людина може в будь якому випадку.

* 1. **Основний код нашої програми**

using System;

using System.Drawing;

using System.Windows.Forms;

namespace RingwormSimulation

{

static class Program

{

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new MainForm());

}

}

public class MainForm : Form

{

private const int n = 15; // Розмірність ділянки шкіри (непарне число)

private const double infectionProbability = 0.5;

private const int timeSteps = 20; // Кількість інтервалів часу для моделювання

private enum CellState

{

Healthy,

Infected,

Immune

}

private CellState[,] grid = new CellState[n, n];

private int[,] infectionTime = new int[n, n];

private Timer timer = new Timer();

private int currentTimeStep = 0;

public MainForm()

{

this.Text = "Ringworm Simulation";

this.Size = new Size(n \* 40, n \* 40);

this.Paint += new PaintEventHandler(this.MainForm\_Paint);

this.Resize += new EventHandler(this.MainForm\_Resize);

InitializeGrid();

timer.Interval = 500; // Інтервал часу в мілісекундах

timer.Tick += new EventHandler(TimerTick);

timer.Start();

}

private void InitializeGrid()

{

for (int i = 0; i < n; i++)

{

for (int j = 0; j < n; j++)

{

grid[i, j] = CellState.Healthy;

infectionTime[i, j] = 0;

}

}

grid[n / 2, n / 2] = CellState.Infected; // Центральна клітинка заражена

}

private void TimerTick(object sender, EventArgs e)

{

currentTimeStep++;

CellState[,] newGrid = (CellState[,])grid.Clone();

for (int i = 0; i < n; i++)

{

for (int j = 0; j < n; j++)

{

if (grid[i, j] == CellState.Infected)

{

infectionTime[i, j]++;

if (infectionTime[i, j] > 6 && infectionTime[i, j] <= 10)

{

newGrid[i, j] = CellState.Immune;

}

else if (infectionTime[i, j] > 10)

{

newGrid[i, j] = CellState.Healthy;

infectionTime[i, j] = 0;

}

else

{

InfectNeighbors(newGrid, i, j);

}

}

else if (grid[i, j] == CellState.Immune && infectionTime[i, j] > 6)

{

infectionTime[i, j]++;

if (infectionTime[i, j] > 10)

{

newGrid[i, j] = CellState.Healthy;

infectionTime[i, j] = 0;

}

}

}

}

grid = newGrid;

this.Invalidate();

if (currentTimeStep >= timeSteps)

{

timer.Stop();

}

}

private void InfectNeighbors(CellState[,] newGrid, int x, int y)

{

int[] dx = { -1, 1, 0, 0 };

int[] dy = { 0, 0, -1, 1 };

for (int k = 0; k < 4; k++)

{

int nx = x + dx[k];

int ny = y + dy[k];

if (nx >= 0 && nx < n && ny >= 0 && ny < n && grid[nx, ny] == CellState.Healthy)

{

if (new Random().NextDouble() < infectionProbability)

{

newGrid[nx, ny] = CellState.Infected;

infectionTime[nx, ny] = 1;

}

}

}

}

private void MainForm\_Paint(object sender, PaintEventArgs e)

{

Graphics g = e.Graphics;

int cellWidth = this.ClientSize.Width / n;

int cellHeight = this.ClientSize.Height / n;

for (int i = 0; i < n; i++)

{

for (int j = 0; j < n; j++)

{

Color color;

switch (grid[i, j])

{

case CellState.Healthy:

color = Color.White;

break;

case CellState.Infected:

color = Color.Red;

break;

case CellState.Immune:

color = Color.Green;

break;

default:

color = Color.White;

break;

}

using (Brush brush = new SolidBrush(color))

{

g.FillRectangle(brush, j \* cellWidth, i \* cellHeight, cellWidth, cellHeight);

g.DrawRectangle(Pens.Black, j \* cellWidth, i \* cellHeight, cellWidth, cellHeight);

}

}

}

}

private void MainForm\_Resize(object sender, EventArgs e)

{

this.Invalidate(); // Перемальовує форму при зміні її розміру

}

}

}

**ВИСНОВКИ**

В даному дослідженні ми вивчили та застосували різноманітні концепції об'єктно-орієнтованого програмування (ООП) у розробці симуляції поширення інфекції за допомогою мови програмування C#. Шляхом моделювання процесу зараження клітинок на ділянці шкіри, ми вивчили і застосували основні принципи ООП, такі як інкапсуляція, успадкування, поліморфізм та абстракція.

Створені нами класи для представлення клітинок та ділянки шкіри дозволили ефективно моделювати поширення інфекції та відображати зміни стану клітинок з плином часу. Результати цього дослідження підтверджують, що об'єктно-орієнтоване програмування є потужним інструментом для моделювання складних систем та процесів, таких як поширення інфекцій.

Наша робота відкриває нові перспективи для подальших досліджень у сфері моделювання поширення інфекційних захворювань та інших подібних процесів. Також цей проект може стати відмінною основою для подальшого розширення функціональності та оптимізації коду, щоб забезпечити ще більш точне та реалістичне моделювання таких процесів.

Результати цього дослідження підтверджують, що об'єктно-орієнтоване програмування є потужним інструментом для моделювання складних систем та процесів, таких як поширення інфекцій. Абстракція дозволила нам узагальнити процес моделювання та робити його більш гнучким та піддаємим до розширення.

**СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ**

1. Документація по мові програмування C# [Електронний ресурс]. – Режим доступу: <https://learn.microsoft.com/en-us/dotnet/csharp/>
2. Об'єктно-орієнтоване програмування [Електронний ресурс]. – Режим доступу: https://shorturl.at/cjzU3
3. Мова програмування C# [Електронний ресурс]. – Режим доступу: https://foxminded.ua/prohramuvannia-na-si/
4. Проект на Git Hub [Електронний ресурс]. – Режим доступу:
5. Документація про інфекцію стригучого лишаю [Електронний ресурс]. – Режим доступу: https://into-sana.ua/enc/striguchij-lishaj/