Алгоритм Ейлера - це алгоритм пошуку шляху в графі, що проходить через кожну вершину графу лише один раз. Застосування цього алгоритму для генерації лабіринту передбачає створення графу, де кожна вершина відповідає комірці лабіринту, а кожний ребро відповідає стіні між комірками.

Для того, щоб застосувати алгоритм Ейлера для генерації лабіринту, потрібно виконати ряд певних операцій:

Створити сітку комірок, в якій буде зберігатися лабіринт. Кожна комірка має чотири стіни: верхню, нижню, ліву та праву. Взяти будь-яку комірку в лабіринті та відмітити її як поточну. Знайти всі сусідні комірки, які ще не були відвідані та видалити стіну між поточною коміркою та випадковою сусідньою коміркою. Це створить прохід між двома комірками в лабіринті. Потім потрібно відмітити сусідню комірку, з якою було створено прохід, як нову поточну комірку та зробити рекурсію, доки всі комірки в лабіринті не будуть відвідані. Цей процес буде продовжуватися до тих пір, поки не буде створено прохід між кожною коміркою в лабіринті, і буде створено повний лабіринт [4].

Отже, лабіринт – це системний сплутаний шлях, який можна вважати графом для подальшої роботи з ним. Для генерування лабіринтів будь-якого розміру та рівня складності можна використовувати графи та алгоритми, які базуються на роботі з ними.

Список літератури:

1. *Baelbung “Algorithm to Generate a Maze”, URL:* https://www.baeldung.com/cs/maze-generation
2. *Geek for geeks “Depth First Search or DFS for a Graph”, URL:* https://www.geeksforgeeks.org/depth-first-search-or-dfs-for-a-graph/
3. *Wikipedia “Prim's algorithm”, URL:* https://en.wikipedia.org/wiki/Prim%27s_algorithm
4. *The Buckblog “Maze Generation: Eller's Algorithm”, URL:* https://weblog.jamisbuck.org/2010/12/29/maze-generation-eller-s-algorithm

УДК 004.9

Шафорост В. В.,
студент 3 курсу спеціальності 122
«Комп’ютерні науки»
Січко Т. В., к. т. н., доцент, доцент
кафедри інформаційних технологій

АЛГОРИТМІЗАЦІЯ ТА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Донецький національний університет імені Василя Стуса

Розробка програмного забезпечення та створення алгоритмів є важливою частиною усіх готових програмних продуктів.

У наш час неможливо уявити діяльність будь-якої успішної організації чи підприємства без ефективного використання інформаційних технологій та інформаційних систем, проектування та розробка яких завершується процесом створення програмних продуктів. Основою програмування є процес алгоритмізації та поняття алгоритму, а також знання алгоритмічних мов [1].

Основною перевагою у розробці програмного забезпечення є висока оплата праці та кількість вакансій. Розробники програмного забезпечення також мають певну гнучкість у своїй роботі. У більшості випадків для виконання повсякденних завдань потрібне лише надійне підключення до Інтернету та комп'ютер. Це часто означає роботу з дому або роботу з будь-якого місця. Проте в розробці програмного забезпечення є й свої недоліки. Найбільш згадуваним недоліком цієї галузі є величезний обсяг роботи, яку доводиться виконати, адже високу популярність можна підтримувати лише тоді, коли є стійкий попит на роботу, і це справді стосується розробників програмного забезпечення. Також постійно з'являються нові мови та інструменти до яких потрібно пристосовуватися для подальшого впровадження і застосування в практичній діяльності [2].

Стратегії компаній щодо розробки програмного забезпечення значно відрізняються залежно від їхніх бізнес-цілей. Інтернет-магазини можуть зосередитися на веб-дизайні та виконанні замовлень, наприклад, тоді як постачальники бізнес-послуг повинні створювати складні системи керування даними. Ось кілька основних категорій розробки програмного забезпечення [3].

Розробка програм. Стосується створення певного комп'ютерного програмного забезпечення для виконання повсякденних функцій у традиційних операційних системах або мобільних пристроях. Це може включати ігри чи інші корисні програми. Мови програмування включають Java, Python або C#. Одним з найбільш ефективних інструментів для створення програм є інтегроване середовище розробки. Воно являє собою набір програмних засобів, що надають зручний інтерфейс для роботи з кодом програми. Прикладом відомої і зручної програми для розробки програм є Visual Studio.

Розробка операційних систем. Включає розробку внутрішніх програм, які забезпечують функціонування операційної системи. Це включає програмне забезпечення для персональних комп'ютерів, бізнес-систем і мобільних телефонів, а також охоплює архітектуру серверів і керування базами даних. Мови включають Java, Python і C/C++. Наприклад, у розробці програмного забезпечення для мікроконтролерів, які використовуються у певних вбудованих системах, мова програмування C є дуже хорошим рішенням. Вона дозволяє ефективно використовувати ресурси, що в свою чергу пришвидшує виконання коду. Саме такі контролери застосовують у різноманітних електронних пристроях, а саме у засобах керування світлофорів або системах клімат-контролю тощо.

Розробка інтерфейсу користувача також відома як UX-дизайн. Цей тип розробки програмного забезпечення створює зовнішні системи, які дозволяють

користувачам взаємодіяти з програмами. Дизайнери займаються візуальною естетикою, функціональними макетами, сумісністю з платформами та виправленням помилок і можуть використовувати мови, зокрема JavaScript і HTML. Прикладами інструментів розробки інтерфейсу користувача можуть бути Axure, Sketch, Adobe XD, Figma тощо. Наприклад, Figma функціонує як платформа розробки веб-інтерфейсу, яка дозволяє користувачам створювати прототипи, змінювати макети, спілкуватися з членами команди та відстежувати зміни в реальному часі [4].

Веб-розробка. Включає в себе розробку та кодування програмного забезпечення та програм для використання у веб-браузерах, включаючи взаємопов'язані файли та сторінки. Мови включають HTML і JavaScript. Наприклад, якщо розробник має намір створити веб-сайт для продажу товарів, він може використати HTML для створення базової структури сторінки, CSS для стилізації та розміщення елементів на сторінці та JavaScript для впровадження динамічних функцій сайту, наприклад анімації, плавні переходи та взаємодія з користувачем. Крім того, вони мають можливість використовувати різні фреймворки та бібліотеки, такі як React або Angular, щоб оптимізувати процес розробки та підвищити продуктивність веб-додатків [5].

Розробка вбудованих систем. Стосується створення операційних систем і програмного забезпечення IoT (Інтернет речей) для некомп'ютерних пристроїв, таких як побутова техніка чи автомобілі. Програми включають Python, Java та Embedded C. Наприклад, розробка вбудованих систем для рентгенівських апаратів та томографів, включаючи програмне забезпечення є прикладом використання мовного програмування Python та Embedded C. Розробка вбудованих систем вимагає оптимізації програмного забезпечення для ефективного функціонування в межах обмеженої ємності пам'яті та обмежень потужності обробки [6].

Розглянемо приклад із повсякденного життя. Можна прокинутися від будильника на телефоні, який підкаже час доби. На роботі можна замовляти каву та закуски для своєї команди за допомогою додатків для доставки їжі, спілкуватися з колегами електронною поштою або в чатах.

Як видно, щоденні звички більшості людей значною мірою залежать від програмного забезпечення. Транспорт, їжа, робота, покупки, побачення, розваги – програмне забезпечення революціонізує майже кожен аспект нашого життя. Це чудова новина для розробників програмного забезпечення, оскільки саме вони створюють програми та системи на які ми так сильно покладаємося.

Список літератури:

1. Розробка програмного забезпечення: веб-сайт URL: http://um.co.ua/7/7-10/7-106137.html (дата звернення 10.04.2023).

2. Розробка програмного забезпечення. Vuzlit: веб-сайт URL: http://surl.li/grvmd (дата звернення 16.04.2023).

3. Степанюк О.С., Січко Т.В. Порівняльний аналіз інструментів для побудови додатків з доповненою реальністю. Комп'ютерні технології обробки даних: матеріали всеукр. наук.-практ. конф., м. Вінниця, 2021. С. 98-101.

4. Розробка програмного забезпечення. Wikipedia: веб-сайт URL: http://surl.li/grvmf (дата звернення 11.04.2023).

5. Січко Т.В., Юрчук Б.О. Розробка веб-додатку туристичної фірми. Вісник Хмельницького національного університету. Технічні науки. №4. 2018. С. 166 - 172.

6. Алгоритмізація на прикладах. Yevshan: веб-сайт URL: http://surl.li/grvmh (дата звернення 13.04.2023).

УДК 519.1

Шевцов М.В., студент 1 курсу
спеціальності 122 «Комп'ютерні науки»
Ніколюк П.К., д.ф-м.н, професор,
професор кафедри інформаційних технологій

ПОРІВНЯННЯ РІЗНИХ ЕВРИСТИЧНИХ ФУНКЦІЙ В АЛГОРИТМІ A*

Донецький національний університет імені Василя Стуса, м. Вінниця

Алгоритм A* є одним за найбільш розповсюджених алгоритмів пошуку оптимального шляху. Для ефективної роботи цього алгоритму необхідно використовувати так звану евристичну функцію, яка оцінює відстань, що залишилась від поточної вершини до цільової. Для A* алгоритму ця функція виглядає наступним чином:

$f(x) = g(x) + h(x)$ [2]

де g(x) – функція вартості досягнення по ребрах графа вершини, яку ми розглядаємо, від начальної а h(x) – функція оцінки відстані від поточної вершини до кінцевої по прямій[1].

Існує декілька евристичних функцій, яку можуть бути використані в алгоритмі A*. Вони відрізняються по способу розрахунку та оцінці відстані, що залишилася. Розглянемо деякі з них.

Евклідова відстань

Евклідова відстань – це відстань між двома точками у просторі. Це найбільш інтуїтивно зрозуміле поняття, яке зустрічається у геометрії.[1] Розраховується наступним чином:

$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$ [1]

В алгоритмі A* евклідова відстань використовується для оцінки відстані між поточною вершиною та цільовою. Це може бути досить ефективним для прямолінійних маршрутів, проте може бути неефективним для маршрутів з перешкодами.

Манхеттенська відстань

Манхеттенська відстань – це евклідова відстань, сітка обмежена сіткою руху, де ми можемо переміщатися тільки у горизонтальному або вертикальному напрямках[1]. Вона розраховується по формулі: