ЗМІСТ

[ВСТУП 5](#_Toc10376311)

[1 ТЕХНІЧНЕ ЗАВДАННЯ 6](#_Toc10376312)

[1.1 Загальні відомості 6](#_Toc10376313)

[1.2 Призначення і цілі створення системи 6](#_Toc10376314)

[1.3 Характеристика об'єктів автоматизації 6](#_Toc10376315)

[1.4 Вимоги до системи 7](#_Toc10376316)

[1.5 Вимоги до видів забезпечення 11](#_Toc10376317)

[1.6 Склад і зміст робіт зі створення системи 13](#_Toc10376318)

[1.7 Порядок контролю і приймання системи 13](#_Toc10376319)

[1.8 Вимоги до складу та змісту робіт з підготовки об'єкта автоматизації до введення системи в дію 15](#_Toc10376320)

[1.9 Вимоги до документування 15](#_Toc10376321)

[1.10 Джерела розробки 16](#_Toc10376322)

[2 АНАЛІЗ ПРОБЛЕМИ 17](#_Toc10376323)

[2.1 Математична модель 17](#_Toc10376324)

[2.2 Графічна бібліотека OpenGL 17](#_Toc10376325)

[3 ОБ'ЄКТНО-ОРІЄНТОВАНИЙ АНАЛІЗ І ПРОЕКТУВАННЯ СИСТЕМИ 20](#_Toc10376326)

[3.1 Бізнес-моделювання предметної області з використанням методологій IDEF0, IDEF3 20](#_Toc10376327)

[3.2 Моделювання систем 21](#_Toc10376328)

[4 ОПИС ПРОГРАМНОГО ПРОДУКТУ 28](#_Toc10376331)

[4.1 Формат вхідних / вихідних даних 28](#_Toc10376332)

[4.2 Модульна структура продукту 28](#_Toc10376333)

[4.3 Документація з програмного коду 28](#_Toc10376334)

[5 ВИПРОБУВАННЯ ПРОГРАМНОГО ПРОДУКТУ 30](#_Toc10376335)

[6 ОПИС ЗАСТОСУВАННЯ 37](#_Toc10376338)

[ВИСНОВКИ 39](#_Toc10376339)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 40](#_Toc10376340)

[ДОДАТОК А 41](#_Toc10376341)

# 

# ВСТУП

Практично в кожному програмному проекті виникає необхідність обробки великого числа однаково організованих даних. У таких випадках подібні дані зручно обробляти як єдине ціле, для чого він представляється у вигляді масиву - іменованої послідовності даних одного типу.

Однією з найпростіших сортувань є бульбашкова сортування. Чи не застосовується для вирішення практичних завдань, через низьку ефективність. Швидко впорядковує тільки масиви дуже невеликого розміру. З іншого боку, метод використовується в навчальних цілях, оскільки лежить в основі деяких інших угруповань, більш ефективних.

Ця курсова робота складається з двох частин: теоретичної та практичної.

Метою теоретичної частини є ознайомлення з алгоритмом бульбашкового сортування, а також з її візуалізацією за допомогою бібліотеки OpenGl.

У практичній частині представлена реалізація бульбашкового сортування і її візуалізація.

Мета курсової роботи: розглянути візуалізацію алгоритму бульбашкового сортування.

Об'єкт дослідження: візуалізація бульбашкового сортування в програмуванні.

Предмет дослідження: масиви в програмуванні.

Актуальність теми: з плином часу всі мови програмування удосконалювалися і отримували нові функції, так само з'являлися і нові методи сортування даних, але є і дуже старі методи угруповань, які використовують і зараз.

# ТЕХНІЧНЕ ЗАВДАННЯ

1.1 Загальні відомості

Повне найменування: Візуалізація алгоритму бульбашкового сортування за допомогою бібліотеки OpenGL.

Скорочене найменування: ВАБС, Система.

Робота виконується на підставі усного договору між Замовником та Розробником.

Замовник: Національний аерокосмічний університет ім. М. Є. Жуковського "Харківський авіаційний інститут", кафедра математичного моделювання та штучного інтелекту

Адреса фактична: г. Харьков, ул. Чкалова №17

Телефон / Факс: +38 066 467 5598

Розробник: Скіцан Ольга Дмитрівна, Лоцкіна Юлія Сергіївна

Адреса фактична: г. Харьков, ул. Чкалова №17

Телефон / Факс: +380671217340

Початок роботи: 17.01.2019

Закінчення роботи: 05.05.2019

Роботи зі створення ВАБС здаються Розробником поетапно відповідно до календарного плану Проекту. Після закінчення кожного з етапів робіт Розробник здає Замовнику відповідні звітні документи етапу, склад яких визначено Договором.

1.2 Призначення і цілі створення системи

ВАБС призначена для сортування масивів випадкових величин з розмірністю більше 100 елементів.

Основним призначенням ВАБС є автоматизація інформаційно-аналітичної діяльності в бізнес-процесах Замовника.

## 1.3 Характеристика об'єктів автоматизації

В Таблиці 1.1 приведені об’єкти автоматизації:

Таблиця 1.1 – Харастеристика об’єктів автоматизації

|  |  |  |  |
| --- | --- | --- | --- |
| Структурний підрозділ | Найменування процесу | Можливість автоматизації | Рішення про автоматизацію в ході проекту |
| Кафедра математичного моделювання та штучного інтелекту | Генерація даних | Можлива | Буде автоматизовано |

Продовження табл. 1.1

|  |  |  |  |
| --- | --- | --- | --- |
| Структурний підрозділ | Найменування процесу | Можливість автоматизації | Рішення про автоматизацію в ході проекту |
| Кафедра математичного моделювання та штучного інтелекту | Візуалізація алгоритму | Можлива | Буде автоматизовано |
| Кафедра математичного моделювання та штучного інтелекту | Сортування даних | Можлива | Буде автоматизовано |
| Кафедра математичного моделювання та штучного інтелекту | Багатопоточність | Можлива | Не буде автоматизовано |

1.4 Вимоги до системи

В [склад персоналу](http://prj-exp.ru/dwh/dwh_project_team.php), необхідного для забезпечення експлуатації ВАБС в рамках відповідних підрозділів Замовника, необхідно виділення наступних відповідальних осіб:

- Розробник ІС - 2 людини.

До [кваліфікації](http://prj-exp.ru/dwh/dwh_team_skills.php) персоналу, що експлуатує Систему ВАБС, висуваються такі вимоги.

- Кінцевий користувач - знання відповідної предметної області; базові навички роботи на персональному комп'ютері.

- Розробник ІС – навички роботи в Microsoft Visual Studio на мові програмування C #; знання і навички необхідні для настройки програмної і апаратної частини системи, для класифікації та усунення виникаючих помилок, і бути ознайомлений з робочою документацією на систему.

Персонал, що працює з Системою ВАБС і виконує функції її супроводу і обслуговування, повинен працювати в наступних режимах:

- Кінцевий користувач – відповідно до основним робочим графіком Замовника.

- Розробник – визначається режимом роботи організації, що експлуатує Систему, за винятком робіт з усунення можливих помилок ПЗ, виявлених в період дослідної експлуатації.

Рівень надійності повинен досягатися узгодженим застосуванням організаційних, організаційно-технічних заходів і програмно-апаратних засобів.

Надійність повинна забезпечуватися за рахунок:

- застосування технічних засобів, системного і базового програмного забезпечення, що відповідають класу вирішуваних завдань;

- своєчасного виконання процесів адміністрування Системи ВАБС; - дотримання правил експлуатації і технічного обслуговування програмно-апаратних засобів;

- попереднього навчання користувачів і обслуговуючого персоналу.

1.4.1 Перелік аварійних ситуацій, за якими регламентуються вимоги до надійності

Під аварійною ситуацією розуміється аварійне завершення процесу, виконуваного тієї чи іншої підсистемою ВАБС, а також «зависання» цього процесу.

При роботі системи можливі наступні аварійні ситуації, які впливають на надійність роботи системи:

- збій в електропостачанні робочої станції користувачів системи;

- помилки Системи ВАБС, не виявлені при налагодженні і випробуванні системи;

- збої програмного забезпечення.

1.4.2 Вимоги до надійності технічних засобів і програмного забезпечення

До надійності обладнання висуваються такі вимоги:

- в якості апаратних платформ повинні використовуватися кошти з підвищеною надійністю;

- застосування технічних засобів відповідних класу вирішуваних завдань;

- апаратно-програмний комплекс Системи повинен мати можливість відновлення у випадках збоїв.

До надійності електропостачання ставляться такі вимоги:

- з метою підвищення відмовостійкості системи в цілому необхідна обов'язкова комплектація серверів джерелом безперебійного живлення з можливістю автономної роботи системи.

Надійність апаратних і програмних засобів повинна забезпечуватися за рахунок наступних організаційних заходів:

- попереднього навчання користувачів і обслуговуючого персоналу;

- своєчасного виконання процесів адміністрування;

- дотримання правил експлуатації і технічного обслуговування програмно-апаратних засобів;

- своєчасне виконання процедур резервного копіювання даних.

Надійність програмного забезпечення підсистем повинна забезпечуватися за рахунок:

- надійності загальносистемного ПЗ та ПЗ, що розробляється Розробником;

- проведенням комплексу заходів налагодження, пошуку і виключення помилок.

1.4.3 Вимоги до методів оцінки і контролю показників надійності на різних стадіях створення системи відповідно до чинних нормативно-технічних документів.

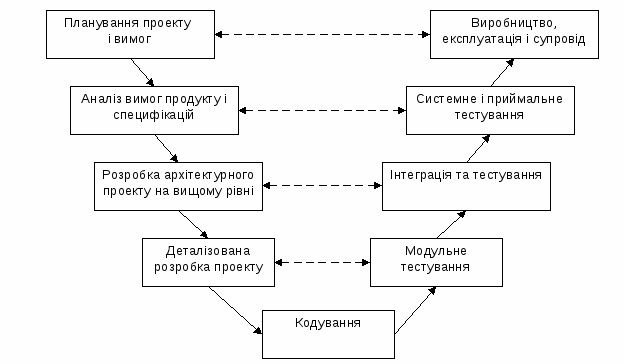


Рисунок 1.1 – V-подібна модель

Для того, щоб контролювати створення проекту будемо слідувати V-моделі представленої на рисунку 1.1.

У моделі на кожному етапі відбувається контроль поточного процесу, для того щоб переконається в можливості переходу на наступний рівень.

Однак недоліком є те, що при виявленні помилок на нижньому рівні, необхідно змінювати всю модель.

Підсистема формування та візуалізації звітності даних повинна забезпечувати зручний для кінцевого користувача інтерфейс, який відповідає таким вимогам.

У частині зовнішнього оформлення:

- інтерфейси підсистем повинен бути типізовані;

- повинен використовуватися шрифт: Monotype Corsiva;

- розмір шрифту повинен бути: 14-20 pt;

У частині діалогу з користувачем:

- при виникненні помилок в роботі підсистеми на екран монітора має виводитися повідомлення з найменуванням помилки і з рекомендаціями щодо її усунення українською мовою.

До інших підсистем ставляться такі вимоги до ергономіки та технічної естетики.

У частині зовнішнього оформлення:

- інтерфейси по підсистемах повинен бути типізовані.

У частині діалогу з користувачем:

- при виникненні помилок в роботі підсистеми на екран монітора має виводитися повідомлення з найменуванням помилки і з рекомендаціями щодо її усунення російською мовою.

1.4.4 Вимоги до експлуатації, технічного обслуговування, ремонту і зберігання компонентів системи

Умови експлуатації, а також види і періодичність обслуговування технічних засобів Системи повинні відповідати вимогам по експлуатації, технічного обслуговування, ремонту і зберігання, викладеним в документації заводу-виготовлювача (виробника) на них.

Технічні засоби Системи і персонал повинні розміщуватися в існуючих приміщеннях Замовника, які за кліматичними умовами повинні відповідати ГОСТ 15150-69 «Машини, прилади та інші технічні вироби. Виконання для різних кліматичних районів. Категорії, умови експлуатації, зберігання і транспортування в частині впливу кліматичних факторів зовнішнього середовища »(температура навколишнього повітря від 5 до 40 ° С, відносна вологість від 40 до 80% при Т = 25 ° С, атмосферний тиск від 630 до 800 мм ртутного стовпа). Розміщення технічних засобів і організація автоматизованих робочих місць повинні бути виконані відповідно до вимог[ГОСТ 21958-76](http://prj-exp.ru/gost/gost_21958-76.php)«Система" Людина-машина ". Зал і кабіни операторів. Взаємне розташування робочих місць. Загальні ергономічні вимоги ».

Для електроживлення технічних засобів повинна бути передбачена трифазна чотирипровідна мережу з глухо заземленою нейтраллю 380/220 В (+10-15)% частотою 50 Гц (+1-1) Гц. Кожне технічне засіб живиться однофазним напругою 220 В частотою 50 Гц через мережеві розетки з заземлюючим контактом.

Для забезпечення виконання вимог по надійності повинен бути створений комплект запасних виробів і приладів (ЗІП).

Склад, місце та умови зберігання ЗІП визначаються на етапі технічного проектування.

1.4.5 Вимоги до захисту інформації від несанкціонованого доступу

Не пред'являються.

1.4.6 Вимоги по стандартизації і уніфікації

У вимоги до стандартизації та уніфікації включають: показники, що встановлюють необхідний ступінь використання стандартних, уніфікованих методів реалізації функцій (завдань) системи, що поставляються програмних засобів, типових математичних методів і моделей, типових проектних рішень, уніфікованих форм управлінських документів, встановлених ГОСТ 6.10.1, загальносоюзних класифікаторів техніко-економічної інформації та класифікаторів інших категорій відповідно до галузі їх застосування, вимоги до використання типових автоматизованих робочих місць, компонентів і комплексів.

Розробка системи повинна здійснюватися з використанням стандартних методологій функціонального моделювання: IDEF0, DFD в рамках рекомендацій по стандартизації Р50.1.028-2001 «Інформаційні технології підтримки життєвого циклу продукції. Методологія функціонального моделювання ».

Моделювання має виконуватися в рамках стандартів, підтримуваних програмними засобами моделювання BPWin.

Для розробки призначених для користувача інтерфейсів і засобів генерації звітів (будь-яких твердих копій) повинні використовуватися вбудовані можливості ПО, а також мова програмування C #.

1.4.7 Вимоги безпеки

У вимоги з безпеки включають вимоги по забезпеченню безпеки при монтажі, наладці, експлуатації, обслуговуванні і ремонті технічних засобів системи (захист від впливів електричного струму, електромагнітних полів, акустичних шумів і т. П.) Щодо допустимих рівнів освітленості, вібраційних і шумових навантажень.

При впровадженні, експлуатації та обслуговуванні технічних засобів системи повинні виконуватися заходи електробезпеки відповідно до «Правил улаштування електроустановок» і «Правилами техніки безпеки при експлуатації електроустановок споживачів».

Апаратне забезпечення системи повинно відповідати вимогам пожежної безпеки в виробничих приміщеннях по ГОСТ 12.1.004-91. «ССБТ. Пожежна безпека. Загальні вимоги".

Повинно бути забезпечено дотримання загальних вимог безпеки відповідно до ГОСТ 12.2.003-91. «ССБТ. Оборудование производственное. Загальні вимоги безпеки »при обслуговуванні системи в процесі експлуатаціі.Аппаратная частина системи повинна бути заземлена відповідно до вимог ГОСТ 50571.22-2000. «Електроустановки будівель. Частина 7. Вимоги до спеціальних електроустановок. Розділ 707. Заземлення обладнання обробки інформації ».

1.5 Вимоги до видів забезпечення

Не пред'являються.

1.5.1 Вимоги до інформаційного забезпечення

Наводяться вимоги:

1. до складу, структури і способів організації даних в системі;
2. до структури процесу збору, обробки, передачі даних в системі і представлення даних;
3. до зберігання даних;
4. до процедури надання юридичної сили документам, продукуються технічними засобами АС (відповідно до [ГОСТ 6.10.4](http://prj-exp.ru/gost/gost_6-10-4-84.php)).

1.5.2 Вимоги до лінгвістичного забезпечення

Для лінгвістичного забезпечення системи наводяться вимоги до застосування в системі мов програмування високого рівня, мов взаємодії користувачів і технічних засобів системи, а також вимоги до кодування і декодування даних, до мов введення-виведення даних, мов маніпулювання даними, засобам опису предметної області (об'єкта автоматизації), до способів організації діалогу.

При реалізації системи повинні застосовуватися такі мови високого рівня: С #.

При реалізації системи повинні застосовуватися такі мови і стандарти взаємодії ВАБС із суміжними системами і користувачів з ВАБС.

Для опису предметної області (об'єкта автоматизації) повинен використовуватися BPWin.

Для організації діалогу системи з користувачем повинен застосовуватися графічний віконний призначений для користувача інтерфейс.

1.5.3 Вимоги до програмного забезпечення

Перелік покупних програмних засобів:

- Microsoft Visual Studio 2013.

Microsoft Visual Studio 2013 повинна мати можливість установки на ОС Windows 7,10.

До забезпечення якості ВАБС ставляться такі вимоги:

- функціональність повинна забезпечуватися виконанням підсистемами всіх їх функцій.

- надійність повинна забезпечуватися за рахунок попередження помилок - недопущення помилок в готових ВАБС;

- легкість застосування повинна забезпечуватися за рахунок застосування покупних програмних засобів;

- ефективність повинна забезпечуватися за рахунок прийняття відповідних, вірних рішень на різних етапах розробки ВАБС і системи в цілому;

- сопровождаемость повинна забезпечуватися за рахунок високої якості документації по супроводу, а також за рахунок використання в програмному тексті опису об'єктів і коментарів; використанням осмислених (мнемонічних) і стійко помітних імен об'єктів; розміщенням не більш одного оператора в рядку тексту програми; униканням створення фрагментів текстів програм з неочевидним або прихованим змістом.

- також на кожному етапі в розробці ВАБС повинна проводиться перевірка правильності прийнятих рішень щодо розробки та застосування готових ВАБС.

Необхідність узгодження розроблених нових програмних засобів з фондом алгоритмів і програм відсутній.

1.5.4 Вимоги до технічного забезпечення

1. Мінімальні вимоги:
2. Процесор з частотою 1,6 ГГц
3. 1 ГБ оперативної пам'яті
4. 4 ГБ вільного місця на жорсткому диску 5400 об / хв
5. Рекомендовані вимоги
6. Процесор з частотою 2 ГГц
7. 4 ГБ оперативної пам'яті

1.5.4 Вимоги до метрологічного забезпечення

Не пред'являються.

1.5.5 Вимоги до організаційного забезпечення

До організації функціонування Системи ВАБС і порядку взаємодії персоналу, який забезпечує експлуатацію, і користувачів пред'являються такі вимоги:

- в разі виникнення з боку функціонального підрозділу необхідності зміни функціональності системи ВАБС, користувачі повинні діяти таким чином: повідомити про помилку Розробникові;

1.6 Склад і зміст робіт зі створення системи

Роботи зі створення системи виконуються в три етапи:

Проектування. Розробка ескізного проекту. Розробка технічного проекту (тривалість – 1,5 місяць).

Розробка робочої документації. Адаптація програм (тривалість – 1,5 місяця).

Введення в дію (тривалість – 1 місяць).

Конкретні терміни виконання стадій і етапів розробки і створення Системи визначаються Планом виконання робіт, який є невід'ємною частиною Договору на виконання робіт по справжньому Приватному технічним завданням.

Перелік організацій – виконавців робіт, визначення відповідальних за проведення цих робіт організацій визначаються Договором.

1.7 Порядок контролю і приймання системи

1.7.1 Види і обсяг випробувань системи

Система піддається випробуванням наступних видів:

1. Попередні випробування.

2. Дослідна експлуатація.

3. Приймальні випробування.

1.7.2 Вимоги до приймання робіт за стадіями

Вимоги до приймання робіт по стадіях наведені в Таблиці 1.2.

Таблиця 1.2 - Вимоги до прийняття робіт по стадіях

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Стадії розробки | Етапи робіт | Зміст робіт | Терміни | Відповідальний виконавець |
| 1 | Аналіз вимог | Написання Технічного завдання | Формування завдання | 17.01.2019 – 30.01.2019 | Лоцкіна Юлія Сергіївна, Скіцан Ольга Дмитрівна |
| Розробка user cases | Формування функціональних вимог |
| 2 | Проектування | Вибір методу рішення, побудова алгоритму | Розробка структури вхідних та вихідних даних, уточнення методів | 30.01.2019 –21.02.2019 | Лоцкіна Юлія Сергіївна, Скіцан Ольга Дмитрівна |
|  |  | Підготовка презентації та доповіді |  |  |  |
| 3 | Розробка | Кодування та програмування | Створення текстів як модулів | 21.02.2019 –04.04.2019 | Лоцкіна Юлія Сергіївна, Скіцан Ольга Дмитрівна |
| Модульні тестування | Розробка плану тестування |
| Оформлення документації | Оформлення по частинах |
| 4 | Тестування | Розробка тест-плану |  | 05.04.2019 –20.04.2019 | Лоцкіна Юлія Сергіївна, Скіцан Ольга Дмитрівна |
| Проведення випробувань | Випробування і коригування ПЗ |
| Оформлення протоколів тестування | Оформлення результатів тестів |
| 5 | Впровадження | Оформлення пояснювальної записки |  | 21.04.2019 –13.05.2019 | Лоцкіна Юлія Сергіївна, Скіцан Ольга Дмитрівна |

1.8 Вимоги до складу та змісту робіт з підготовки об'єкта автоматизації до введення системи в дію

Для створення умов функціонування ВАБС, при яких гарантується відповідність створюваної системи вимогам, які містяться в цьому технічному завданні, і можливість ефективного її використання, в організації Замовника повинен бути проведений комплекс заходів.

Перелік регламентів може бути змінений на стадії «Розробка робочої документації. Адаптація програм ».

1.9 Вимоги до документування

Таблиця 1.3 – Вимоги до документування

|  |  |
| --- | --- |
| Етап | Документ |
| Проектування. Розробка ескізного проекту. Розробка технічного проекту. | [Пояснювальна записка до технічного проекту](http://prj-exp.ru/patterns/pattern_tech_project.php) |
| [Схема функціональної структури](http://prj-exp.ru/patterns/diagram_functional_structure.php) |
| Розробка робочої документації. адаптація програм | Загальний опис системи |
|  |
| Технологічна інструкція |
| [Інструкція користувача](http://prj-exp.ru/patterns/pattern_user_guide.php) |
| Опис технологічного процесу обробки даних (включаючи телеобробки) |
| Склад вихідних даних (повідомлень) |
| [Програма](http://prj-exp.ru/patterns/pattern_program_of_test.php) і [методика випробувань](http://prj-exp.ru/patterns/pattern_methods_of_test.php) |
| Специфікація |
| Опис програм |
| Текст програм |
| Введення в дію | [Акт приймання в дослідну експлуатацію](http://prj-exp.ru/patterns/pattern_act_of_trial_operation.php) |
| [Протокол випробувань](http://prj-exp.ru/patterns/pattern_report_of_test.php) |
| Акт завершення робіт |

Вся документація повинна бути підготовлена і передана як в друкованому, так і в електронному вигляді (у форматі Microsoft Word).

1.10 Джерела розробки

Технічне завдання розроблено на основі наступних документів та інформаційних матеріалів:

- ГОСТ 24.701-86 «Надійність автоматизованих систем управління».

- ГОСТ 15150-69 «Машини, прилади та інші технічні вироби. Виконання для різних кліматичних районів. Категорії, умови експлуатації, зберігання і транспортування в частині впливу кліматичних факторів зовнішнього середовища ».

-[ГОСТ 21958-76](http://prj-exp.ru/gost/gost_21958-76.php)«Система «Людина-машина». Зал і кабіни операторів. Взаємне розташування робочих місць. Загальні ергономічні вимоги ».

- ГОСТ 12.1.004-91 «ССБТ. Пожежна безпека. Загальні вимоги».

- ГОСТ 50571.22-2000 «Електроустановки будівель».

- і т.д.

# АНАЛІЗ ПРОБЛЕМИ

## Математична модель

Сортування бульбашкою ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) bubble sort) – простий [алгоритм сортування](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D1%81%D0%BE%D1%80%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%BA%D0%B8). Для розуміння і реалізації цей алгоритм – найпростіший, але ефективний він лише для невеликих масивів.

Алгоритм складається з повторюваних проходів по сортованого масиву. За кожен прохід елементи послідовно порівнюються попарно і, якщо порядок в парі невірний, виконується обмін елементів. Проходи по масиву повторюються раз або до тих пір, поки на черговому проході не опиниться, що обміни більше не потрібні, що означає - масив відсортований. При кожному проході алгоритму по внутрішньому циклу, черговий найбільший елемент масиву ставиться на своє місце в кінці масиву поруч з попереднім «найбільшим елементом», а найменший елемент переміщається на одну позицію до початку масиву («спливає» до потрібної позиції, як бульбашка у воді - звідси і назва алгоритму).

Модифікації бульбашкового сортування:

* [коктейльна сортування](http://algolab.valemak.com/cocktail).
* [Парно-непарна сортування](http://algolab.valemak.com/odd-even). За один прохід непарні індекси порівнюються з парними, потім парні - з непарними.
* [Сортування гребінцем](http://algolab.valemak.com/comb). Порівнювати не сусіди, а елементи між якими деяку відстань, меншу з кожним прогоном по масиву.

## Графічна бібліотека OpenGL

Для візуалізації сортування бульбашкою використано графічну бібліотеку OpenGL.

OpenGL – ([англ.](https://uk.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D1%96%D0%B9%D1%81%D1%8C%D0%BA%D0%B0_%D0%BC%D0%BE%D0%B2%D0%B0) Open Graphics Library – відкрита графічна бібліотека)  – специфікація, що визначає незалежний від [мови програмування](https://uk.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B2%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F) [крос-платформовий](https://uk.wikipedia.org/wiki/%D0%9A%D1%80%D0%BE%D1%81-%D0%BF%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%BE%D0%B2%D0%B8%D0%B9) програмний інтерфейс ([API](https://uk.wikipedia.org/wiki/API)) для написання [застосунків](https://uk.wikipedia.org/wiki/%D0%97%D0%B0%D1%81%D1%82%D0%BE%D1%81%D1%83%D0%BD%D0%BE%D0%BA), що використовують [2D](https://uk.wikipedia.org/wiki/2D) та [3D](https://uk.wikipedia.org/wiki/3D) [комп'ютерну графіку](https://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%27%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0_%D0%B3%D1%80%D0%B0%D1%84%D1%96%D0%BA%D0%B0). Цей інтерфейс містить понад 250 функцій, які можуть використовуватися для малювання складних тривимірних сцен з простих [примітивів](https://uk.wikipedia.org/w/index.php?title=%D0%9F%D1%80%D0%B8%D0%BC%D1%96%D1%82%D0%B8%D0%B2&action=edit&redlink=1). Широко застосовується індустрією [комп'ютерних ігор](https://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%27%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D1%96_%D1%96%D0%B3%D1%80%D0%B8) і [віртуальної реальності](https://uk.wikipedia.org/wiki/%D0%92%D1%96%D1%80%D1%82%D1%83%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0_%D1%80%D0%B5%D0%B0%D0%BB%D1%8C%D0%BD%D1%96%D1%81%D1%82%D1%8C), у графічних інтерфейсах ([Compiz](https://uk.wikipedia.org/wiki/Compiz), [Clutter](https://uk.wikipedia.org/wiki/Clutter)), при візуалізації наукових даних, в системах автоматизованого проектування тощо.

OpenGL орієнтується на такі два завдання:

1. Сховати складності адаптації різних 3D-прискорювачів, надаючи розробнику єдиний API.
2. Приховати відмінності в можливостях апаратних платформ, вимагаючи реалізації відсутньої функціональності за допомогою програмної емуляції.

Основним принципом роботи OpenGL є отримання наборів векторних графічних примітивів у вигляді точок, ліній та багатокутників з наступною математичною обробкою отриманих даних та побудовою растрової картинки на екрані і/або в пам'яті. Векторні трансформації та растеризація виконуються графічним конвеєром (graphics pipeline), який власне являє собою дискретний автомат. Абсолютна більшість команд OpenGL потрапляють в одну з двох груп: або вони додають графічні примітиви на вхід в конвеєр, або конфігурують конвеєр на різне виконання трансформацій.

OpenGL є низькорівневим процедурним API, що змушує програміста диктувати точну послідовність кроків, щоб побудувати результуючу растрову графіку (імперативний підхід). Це є основною відмінністю від дескрипторних підходів, коли вся сцена передається у вигляді структури даних (найчастіше дерева), яке обробляється і будується на екрані. З одного боку, імперативний підхід вимагає від програміста глибокого знання законів тривимірної графіки та математичних моделей, з іншого боку — дає свободу впровадження різних інновацій.

OpenGL обслуговує дві цілі:

– Для того, щоб приховувати складнощі встановлення зв'язку комп'ютера з різними [3D акселераторами](https://uk.wikipedia.org/w/index.php?title=3D_%D0%B0%D0%BA%D1%81%D0%B5%D0%BB%D0%B5%D1%80%D0%B0%D1%82%D0%BE%D1%80&action=edit&redlink=1), надати [програмістові](https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%96%D1%81%D1%82) один, загальноприйнятий [API](https://uk.wikipedia.org/wiki/API)

* Для того, щоб приховати можливості базових інструментальних машин, які відрізняються своїм намаганням виконати підтримку повного набору особливостей OpenGL (використання програмної [емуляції](https://uk.wikipedia.org/wiki/%D0%95%D0%BC%D1%83%D0%BB%D1%8F%D1%86%D1%96%D1%8F), якщо необхідно).

Для генерації випадкових чисел використувують клас Random, щоб більш детально ознайомитись з ним, ми використували [1]:

Для того щоб отримати цілі випадкові числа в C #, необхідно оголосити

змінну типу Random.

Псевдовипадкові числа вибираються з однаковою ймовірністю з кінцевого набору чисел. Вибрані числа не є абсолютно випадковими, оскільки для їх вибору використовується математичний алгоритм, але вони є досить випадковими для практичних цілей. Поточна реалізація класу [Random](https://docs.microsoft.com/en-us/dotnet/api/system.random?view=netframework-4.8) ґрунтується на модифікованій версії алгоритму генератора випадкових чисел Дональда Е. Кнута.

Ви створюєте генератор випадкових чисел, надаючи конструктору [випадкового](https://docs.microsoft.com/en-us/dotnet/api/system.random.-ctor?view=netframework-4.8) класу початкове значення (початкове значення для алгоритму генерації псевдовипадкових чисел). Ви можете задати початкове значення явно або неявно:

–конструктор [Random (Int32)](https://docs.microsoft.com/en-us/dotnet/api/system.random.-ctor?view=netframework-4.8#System_Random__ctor_System_Int32_) використовує явне значення, яке ви надаєте.

–конструктор [Random ()](https://docs.microsoft.com/en-us/dotnet/api/system.random.-ctor?view=netframework-4.8#System_Random__ctor) використовує системний годинник, щоб надати початкове значення. Це найбільш поширений спосіб створення генератора випадкових чисел.

Якщо одне і те ж насіння використовується для окремих [випадкових](https://docs.microsoft.com/en-us/dotnet/api/system.random?view=netframework-4.8) об'єктів, вони генерують одну і ту ж серію випадкових чисел. Це може бути корисним для створення тестового набору, який обробляє випадкові значення, або для відтворення ігор, які отримують свої дані з випадкових чисел. Однак слід враховувати, що [випадкові](https://docs.microsoft.com/en-us/dotnet/api/system.random?view=netframework-4.8) об'єкти в процесах, що виконуються в різних версіях .NET Framework, можуть повертати різні серії випадкових чисел, навіть якщо вони створені з ідентичними значеннями насіння.

Щоб виробляти різні послідовності випадкових чисел, можна зробити залежне від часу насіннєве значення, тим самим виробляючи різні серії з кожним новим екземпляром [Random](https://docs.microsoft.com/en-us/dotnet/api/system.random?view=netframework-4.8). Параметризований конструктор [Rand m (Int32)](https://docs.microsoft.com/en-us/dotnet/api/system.random.-ctor?view=netframework-4.8#System_Random__ctor_System_Int32_) може приймати значення [Int32,](https://docs.microsoft.com/en-us/dotnet/api/system.int32?view=netframework-4.8) засноване на кількості тиків у поточний час, тоді як конструктор [Random () без](https://docs.microsoft.com/en-us/dotnet/api/system.random.-ctor?view=netframework-4.8#System_Random__ctor) параметрів використовує системний годинник для генерації його насіннєвого значення. Однак, оскільки годинник має кінцеву роздільну здатність, використовуючи безпараметричний конструктор для створення різних [випадкових](https://docs.microsoft.com/en-us/dotnet/api/system.random?view=netframework-4.8) об'єктів у тісній послідовності створюються генератори випадкових чисел, які виробляють ідентичні послідовності випадкових чисел. Наступний приклад ілюструє два [випадкових](https://docs.microsoft.com/en-us/dotnet/api/system.random?view=netframework-4.8) об'єкти, створені в тісній послідовності, генерують однакові серії випадкових чисел.

# ОБ'ЄКТНО-ОРІЄНТОВАНИЙ АНАЛІЗ І ПРОЕКТУВАННЯ СИСТЕМИ

Для реалізації поставленого завдання найбільш ефективно використовувати об'єктно-орієнтований підхід.

В якості мови програмування шляхом проведення порівняльного аналізу методом варіантних мереж була вибрана мова С #, який підтримує всі необхідні об'єктно-орієнтовані можливості, а також вельми гнучкий. Як середовище розробки під платформу Windows була обрана візуальне середовище розробки Visual Studio 2013. Для проектування системи були використані BPWin і Rational Rose.

## Бізнес-моделювання предметної області з використанням методологій IDEF0, IDEF3

IDEF0 – [методологія](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D1%8F) функціонального моделювання ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) function modeling) і графічна нотація, призначена для формалізації і опису [бізнес-процесів](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%B7%D0%BD%D0%B5%D1%81-%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81). Відмінною особливістю IDEF0 є її акцент на підпорядкованість об'єктів.

Стандарт IDEF0 представляє організацію як набір модулів, тут існує правило – найбільш важлива функція знаходиться у верхньому лівому кутку, крім того є правило боку:

‑ стрілка входу завжди приходить в ліву кромку активності,

‑ стрілка управління – в верхню кромку,

‑ стрілка механізму – нижня кромка,

‑ стрілка виходу – права кромка.

IDEF0 для даної курсової роботи зображена на рисунку 3.1.

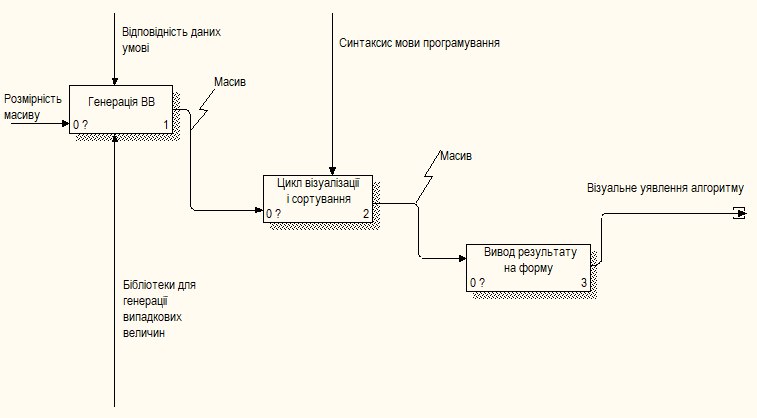


Рисунок 3.1 – Діаграма IDEF0

IDEF3 ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) Integrated DEFinition for Process Description Capture Method) – методологія [моделювання](https://ru.wikipedia.org/w/index.php?title=%D0%91%D0%B8%D0%B7%D0%BD%D0%B5%D1%81-%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5&action=edit&redlink=1) і стандарт документування процесів, що відбуваються в системі. Метод документування технологічних процесів є механізм документування та збору інформації про процеси. IDEF3 показує причинно-наслідкові зв'язки між ситуаціями і подіями в зрозумілій експерту формі, використовуючи структурний метод вираження знань про те, як функціонує система, процес або підприємство.

IDEF3 широко застосовується при розробці інформаційних систем. При цьому використовується інструмент візуального моделювання бізнес-процесів [2].

IDEF3 представлена на рисунку 3.2

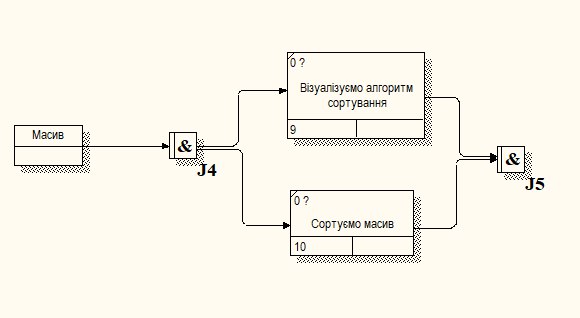


Рисунок 3.2 – IDEF3 діаграма

## Моделювання систем

Щоб більш детально розглянути відомості про моделювання використовуємо [3]:

Заміщення одного об'єкта іншим з метою отримання інформації про найважливіші властивості об'єкта-оригіналу за допомогою об'єкта-моделі називається моделюванням. Таким чином, моделювання може бути визначене як подання об'єкта моделлю для отримання інформації про цей об'єкт шляхом проведення експериментів з його моделлю. Теорія заміщення одних об'єктів (оригіналів) іншими об'єктами (моделями) і дослідження властивостей об'єктів на їх моделях називається теорією моделювання.

Узагальнено моделювання можна визначити як метод опосередкованого пізнання, при якому досліджуваний об'єкт-оригінал знаходиться в деякому відповідно до іншим об'єктом-моделлю, причому модель здатна в тому чи іншому відношенні заміщати оригінал на деяких стадіях пізнавального процесу. Стадії пізнання, на яких відбувається така заміна, а також форми відповідності моделі і оригіналу можуть бути різними:

1. моделювання як пізнавальний процес, що містить переробку інформації, що надходить із зовнішнього середовища, про що відбуваються в ній явища, в результаті чого в свідомості з'являються образи, відповідні об'єктах;
2. моделювання, що полягає в побудові деякої системи-моделі (другий системи), пов'язаної певними співвідношеннями подоби з системою-оригіналом (першої системою), причому в цьому випадку відображення однієї системи в іншу є засобом виявлення залежностей між двома системами, відображеними у співвідношеннях подоби, а не результатом безпосереднього вивчення інформації, що надходить.

### Діаграма прецедентів

Для того, щоб більш детально ознайомитися з діаграмами прецендентов використовуємо [2]:

Ідея опису функціональних вимог у вигляді варіантів використання (use case) була сформульована в 1986 р Іваром Якобсоном. Ця ідея була визнана конструктивною і отримала широке схвалення. Згодом найбільш значний внесок у вирішення проблеми визначення сутності варіантів використання і способів їх опису вніс Алістер Коберн.

Варіант використання являє собою послідовність дій (транзакцій), виконуваних системою у відповідь на подію, що ініціюється деяким зовнішнім об'єктом (дійовою особою). Варіант використання описує типове взаємодія між користувачем і системою і відображає уявлення про поведінку системи з точки зору користувача. У найпростішому випадку варіант використання визначається в процесі обговорення з користувачем тих функцій, які він хотів би реалізувати, або цілей, які він переслідує по відношенню до розроблюваної системі.

Дійова особа (actor) – це роль, яку користувач грає по відношенню до системи. Дійові особи представляють собою ролі, а не конкретних людей або найменування робіт.

Незважаючи на те що на діаграмах варіантів використання вони зображуються у вигляді стилізованих людських фігурок, дійова особа може також бути зовнішньою системою, якій необхідна деяка інформація відданої системи.

Дійові особи діляться на три основних типи – користувачі системи, інші системи, які взаємодіють з даною, і час. Час стає дійовою особою, якщо від нього залежить запуск будь-яких подій в системі.

Дійові особи можуть грати різні ролі по відношенню до варіанту використання. Вони можуть користуватися його результатами самі безпосередньо в ньому брати участь. Значимість різних ролей дійової особи залежить від того, яким чином використовуються його зв'язку.

Мета побудови діаграм варіантів використання – документування функціональних вимог до системи в найзагальнішому вигляді, тому вони повинні бути гранично простими.

При побудові діаграм варіантів використання потрібно дотримуватися наступних правил:

‑ Чи не моделюйте зв'язку між діючими особами. За визначенням дійові особи знаходяться поза сферою дії системи. Це означає, що зв'язки між ними також не належать до її компетенції.

‑ Чи не поєднуйте стрілкою два варіанти використання безпосередньо. Діаграми даного типу описують тільки самі варіанти використання, а не порядок їх виконання. Для відображення порядку виконання варіантів використання застосовують діаграми діяльності.

‑ Кожен варіант використання повинен бути ініційований дійовою особою. Це означає, що завжди повинна бути стрілка, що починається на діючу особу, яка закінчується на варіанті використання.

Хорошим джерелом для ідентифікації варіантів використання служать зовнішні події. Слід почати з перерахування всіх подій, що відбуваються в зовнішньому світі, на які система повинна якимось чином реагувати. Будь-яке конкретне подія може спричинити за собою реакцію системи, яка потребує втручання користувачів, або, навпаки, викликати чисто призначену для користувача реакцію. Ідентифікація подій, на які необхідно реагувати, допомагає ідентифікувати варіанти використання.

Діаграма варіантів використання є найбільш загальним поданням функціональних вимог до системи. Однак моделювання варіантів використання не зводиться тільки до малювання діаграм. Для подальшого проектування системи потрібні більш конкретні деталі. Ці деталі описуються в документі, званому «сценарій варіанта використання» або «потік подій» (flow of events). Метою потоку подій є докладний документування процесу взаємодії дійової особи з системою, що реалізується в рамках варіанту використання. У потоці подій має бути описано все, що служить задоволенню запитів дійових осіб.

Хоча потік подій і описується докладно, він також не повинен залежати від реалізації. Мета – описати, що буде робити система, а не як вона буде робити це. Зазвичай опис потоку подій включає наступні розділи:

‑ короткий опис;

‑ передумови (pre-conditions);

‑ основний потік подій;

‑ альтернативні потоки подій;

‑ постумови (post-conditions);

‑ розширення (extensions).

Переваги моделі варіантів використання полягають в тому, що вона:

‑ визначає користувачів і кордони системи;

‑ визначає системний інтерфейс;

‑ зручна для спілкування користувачів з розробниками;

‑ використовується для написання тестів;

‑ є основою для написання документації користувача;

‑добре вписується в будь-які методи проектування (як об'єктно-орієнтовані, так і структурні).

Для даної курсової роботи була зроблена діаграма прецедентів, яка представлена на рисунку 3.3 та опис потоків подій.

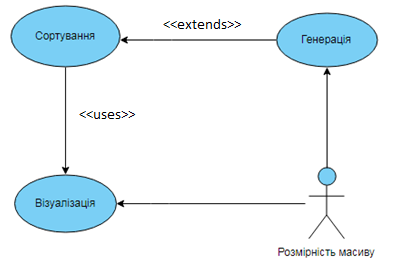


Рисунок 3.3 – Діаграма прецендентів

Опис варіантів використання:

1. Генерация
2. Передумова:

Отримуємо розмірність масиву.

1. Основний потік:

Відбувається генерація масиву, а також перевірка даних на те, що дані не є відсортованими

1. Альтернативний потік:

Частина даних є відсортованою

- Відбувається повторна генерація

1. Постумова:

Згенерований масив з невідсортованими елементами

1. Сортування
2. Передумова:  
   Є згенерований масив
3. Основний потік подій :  
   Відбувається бульбашкове сортування
4. Постумова :  
   Відсортований масив
5. Візуалізація
6. Передумова:  
   Є відсортований масив і його розмірність
7. Основний потік подій :  
   Відбувається візуалізація алгоритму на основі:  
   - Відсортованого масиву   
   - Розмірність масиву
8. Постумова:  
   Візуалізація на основі бульбашкового сортування.

### Діаграма станів

Діаграма станів взаємно пов’язує події та стани. У разі приймання події наступний стан системи залежить як від поточного стану, так і від події. Зміна стану називається переходом [4].

Діаграма станів для даної роботи зоображена на рисунку 3.4

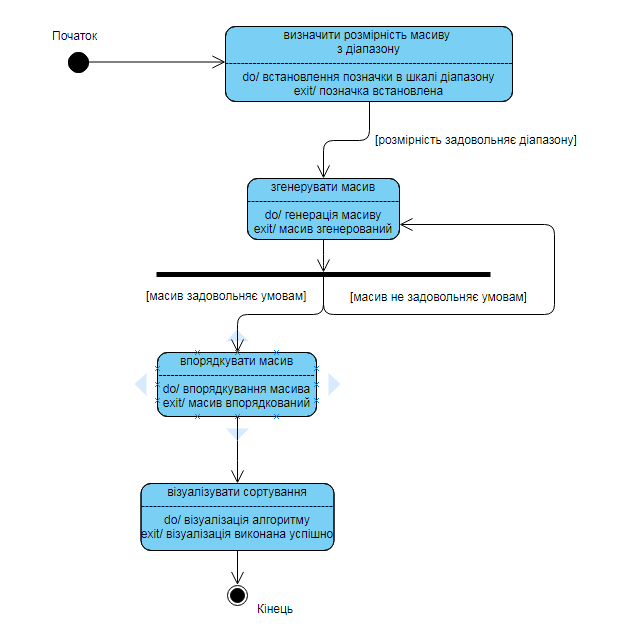


Рисунок 3.4 – Діаграма класів

Діаграма класів ([англ.](https://uk.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D1%96%D0%B9%D1%81%D1%8C%D0%BA%D0%B0_%D0%BC%D0%BE%D0%B2%D0%B0) class diagram) – статичне представлення структури моделі. Відображає статичні (декларативні) елементи, такі як: [класи](https://uk.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B0%D1%81_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)), [типи даних](https://uk.wikipedia.org/wiki/%D0%A2%D0%B8%D0%BF_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85), їх зміст та відношення. Діаграма класів, також, може містити позначення для пакетів та може містити позначення для вкладених пакетів. Також, діаграма класів може містити позначення деяких елементів поведінки, однак їх динаміка розкривається в інших типах діаграм. Діаграма класів служить для представлення статичної структури моделі системи в термінології класів об'єктно-орієнтованого програмування. На цій діаграмі показують класи, інтерфейси, об'єкти й кооперації, а також їхні відносини.

Існує чотири типи зв'язків в UML:

‑ Залежність

‑ Асоціація

‑ Узагальнення

‑ Реалізація

Залежність – семантично представляє собою зв'язок між двома елементами моделі, в якій зміна одного елемента (незалежного) може привести до зміни семантики іншого елемента (залежного). Графічно представлена ​​пунктирною лінією (рисунок 3.5), іноді зі стрілкою, спрямованої до тієї сутності, від якої залежить ще одна; може бути забезпечена міткою.



Рисунок 3.5 – Графічне представлення залежності

Залежність – це зв'язок використання, яка вказує, що зміна специфікацій однієї сутності може вплинути на інші сутності, які використовують її.

Асоціація – це структурна зв'язок між елементами моделі, яка описує набір зв'язків, що існують між об'єктами. Асоціація показує, що об'єкти однієї сутності (класу) пов'язані з об'єктами іншої сутності таким чином, що можна переміщатися від об'єктів одного класу до іншого.

Агрегація – особливий різновид асоціації, що представляє структурну зв'язок цілого з його частинами. Як тип асоціації, агрегація може бути іменованої. Одне ставлення агрегації не може включати більше двох класів (контейнер і вміст). Агрегація зустрічається, коли один клас є колекцією або контейнером інших. Причому, за замовчуванням агрегацией називають агрегацію за посиланням, тобто коли час існування містяться класів не залежить від часу існування містить їх класу. Якщо контейнер буде знищений, то його вміст - немає.

Графічно агрегація представляється порожнім ромбом на блоці класу «ціле», і лінією, що йде від цього ромба до класу «частина» (рисунок 3.6).



Рисунок 3.6 – Графічне представлення агрегації

Композиція – більш суворий варіант агрегації. Відома також як агрегація за значенням. Композиція – це форма агрегації з чітко вираженими відносинами володіння і збігом часу життя частин і цілого. Композиція має жорстку залежність часу існування примірників класу контейнера і примірників містяться класів. Якщо контейнер буде знищений, то весь його вміст буде також знищено.

Графічно представляється як і агрегація, але з зафарбовані ромбиком (рисунок 3.7).



Рисунок 3.7 – Графічне представлення композиції

Третій зв'язок – узагальнення – висловлює спеціалізацію або успадкування, в якому спеціалізований елемент (нащадок) будується за специфікаціями узагальненого елемента (одного з батьків. Графічно узагальнення представлено у вигляді суцільної лінії з порожньою стрілкою, що вказує на батька (рисунок 3.8).



Рисунок 3.8 – Графічне представлення узагальнення

Четвертий – реалізація – це семантична зв'язок між класами, коли один з них (постачальник) визначає угоду, якого другий (клієнт) зобов'язаний дотримуватися.  У графічному виконанні зв'язок реалізації - це гібрид зв'язків узагальнення і залежності: трикутник вказує на постачальника, а другий кінець пунктирною лінії - на клієнта (рисунок 3.9).



Рисунок 3.9 – Графічне представлення реалізації

Діаграма класів для сортування масиву та візуалізації алгоритму бульбашкового сортування зображена на рисунку 3.10

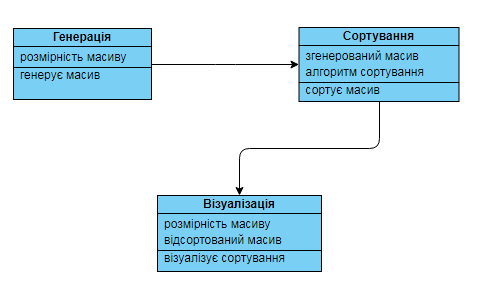


Рисунок 3.10 – Діаграма класів

# ОПИС ПРОГРАМНОГО ПРОДУКТУ

## Формат вхідних / вихідних даних

Формат вхідних даних: кількість елементів в масиві, яка вибирається з заданого діапазону можливої розмірності. Розмірність масиву варіюється від 10 до 50 елементів.

Формат вихідних даних: відсортований масив згенерованих даних і візуальне уявлення сортування. Мінімальним значенням в масиві може бути 5, а максимльним – 175.

## Модульна структура продукту

На рисунку 4.1 приведена модульна структура програмного продукту у вигляді діаграми компонентів на мові UML.

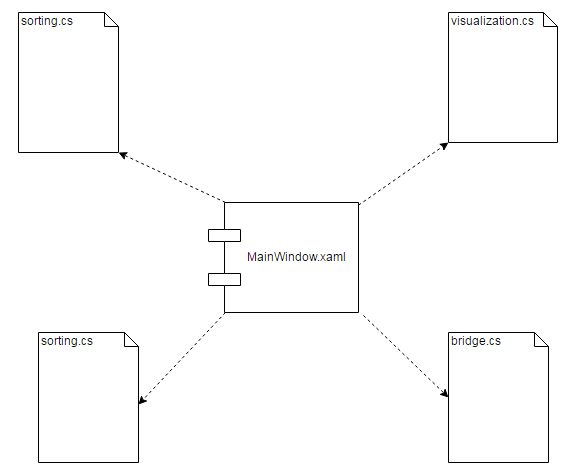


Рисунок 4.1 – Модульна структура програмного продукту

## Документація з програмного коду

Програма містить клас sorting.cs. Цей клас містить два метода:

‑ static public int[] ArrayGeneration(int N) – метод, який генерує масив випадкових значень в діапазоні від 5 до 175. Генерація відбувається завдяки класу Random. Кількість елементів визначає змінна типу int N;

‑ static public int[] BulbSort(int[] array, Line[] lines,Label[]labels) – метод, який містить алгоритм бульбашкового сортування.

Також в програмі є клас visualization.cs. В цьому класі прописані:

‑ static public Label[] InitializeLabel(int[] array) – метод, для зображення ярликів в яких знаходяться елементи згенерованого масиву;

‑ static public Line[] InitializeLines(int[] array) – метод, для зображення паличок;

‑ static public void SwapLines (int index1, int index2 , Line[] lines , int[] array) – метод, для зміни положення двух паличок;

‑ static public void PrintArray (int[] array, Label[] labels) – метод, для друку масива значень.

Для встановлення кольору паличок написан клас color.cs. Клас містить:

‑ static public void SetPurple (int index1, Line[] lines) – метод, для зміни кольору паличок, які вже є упорядкованими;

‑ static public void SetBlue (int index1, int index2, Line[] lines) – метод, для зміни кольору паличок, які перевіряються;

‑ static public void SetRed (int index1, int index2, Line[] lines) – метод, для зміни кольору паличок, які ще не знаходяться на своїх місцях;

Програма містить клас bridge.cs, який містить патерн Мост (Bridge), який є структурним патерном, що розглядає, як класи і об'єкти утворюють більші структури - більш складні за характером класи і об'єкти.

Також програма містить клас client.cs, в якому реалізована ініціалізація паличок і значень, також прописані методи для початку і завершення роботи.

Обробка роботи кнопок міститься в MainWindow.xaml.cs, в файлі реалізований класс Window, який є спадкоємцем класу MainWindow. В класі прописані методи для роботи кнопок та для їх деактивації.

# ВИПРОБУВАННЯ ПРОГРАМНОГО ПРОДУКТУ

1. Ресурси

Для проведення випробувань програмного продукту необхідні такі ресурси.

Персонал: розробники програми.

Вимоги до апаратного забезпечення:

Для роботи необхідний один ПК наступної мінімальної

конфігурації:

* Процесор 533 MHz
* RAM 64 MB
* Free HDD space - 5 MB
* Дисплей, маніпулятор «миша», клавіатура, відеопідсистема.

Для успішної та ефективної роботи необхідна ОС Windows 7/10.

1. Опис test-cases

**TC-1.** Перевірити роботу повзунка «Кількість елементів».

**TC-2.** Перевірити роботу повзунка «Затримка».

**TC-3.** Перевірити роботу кнопки «Генерувати».

**TC-4.** Перевірити роботу кнопки «Сортувати».

**TC-5.** Перевірити роботу програми при виборі максимального значення елементів та максимального значення затримки.

**TC-6.** Перевірити роботу програми при виборі максимального значення елементів та мінімального значення затримки.

**TC-7.** Перевірити роботу програми при виборі мінімального значення елементів та максимального значення затримки.

**TC-8.** Перевірити роботу програми при виборі мінімального значення елементів та мінімального значення затримки.

**TC-9.** Перевірити роботу кнопки «Генерувати» після попереднього сортування.

**TC-10.** Перевірити стан кнопки «Генерувати» під час роботи програми.

**TC-11.** Перевірити стан кнопки «Сортувати» під час роботи програми.

**TC-12.** Перевірити стан повзунка «Кількість елементів» під час роботи програми.

**TC-13.** Перевірити стан повзунка «Затримка» під час роботи програми.

**TC-14.** Перевірити роботу програми при натиснені кнопки «Сортувати» перш ніж написнути кнопку «Генерувати».

У табл. 5.1. представлені результати досліджень зон ризику виникнення помилок.

Таблиця 5. 1 – Зони ризику

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер  Test  case | Номер  відповідний  вимогам ТЗ | Test case | Опис (сценарій) | Очікуваний результат |
| ТС-1 | ТЗ-1.3.3 | Перевірити роботу повзунка «Кількість елементів». | 1.Запуск програмного продукту | 1.Відкриється вікно з програмою |
| 2.Вибрати повзунком кількість елементів | 2.Відображується значення кількості елементів, а повзунок знаходиться в вибраному положені |
| 3.Запустити виконання програми | 3.Програма виконується |
| ТС-2 | ТЗ-1.3.2 | Перевірити роботу повзунка «Затримка». | 1.Запуск програмного продукту | 1.Відкриється вікно з програмою |
| 2.Вибрати повзунком затримку | 2.Відображується значення затримки, а повзунок знаходиться в вибраному положені |
| 3.Запустити виконання програми | 3.Програма виконується |
| ТС-3 | ТЗ-1.3.1 | Перевірити роботу кнопки «Генерувати». | 1.Запуск програмного продукту | 1.Відкриється вікно з програмою |
| 2.Натиснути на кнопку «Генерувати» | 2.Відображення згенерированого масива |
| 3.Запустити виконання програми | 3.Програма виконується |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Продовження табл. 5.1 | | | | |
| Номер  Test  case | Номер  відповідний  вимогам ТЗ | Test case | Опис (сценарій) | Очікуваний результат |
| ТС-4 | ТЗ-1.3.3 | Перевірити роботу кнопки «Сортувати». | 1.Запуск програмного продукту | 1.Відкриється вікно з програмою |
| 2.Натиснути на кнопку «Сортувати» | 2.Візуальне представлення сортування |
| 3.Запустити виконання програми | 3.Програма виконується |
| ТС-5 | ТЗ-1.3.2 | Перевірити роботу програми при виборі максимального значення елементів та максимального значення затримки. | 1.Запуск програмного продукту | 1.Відкриється вікно з програмою |
| 2.Встановлення повзунка «Кількість елементів» та «Затримка» на максимальних позначках | 2.Відображення значень кількості елементів та затримки, повзунки знаходяться на максимальних позначках |
| 3.Запустити виконання програми | 3.Програма виконується |
| ТС-6 | ТЗ-1.3.2 | Перевірити роботу програми при виборі максимального значення елементів та мінімального значення затримки. | 1.Запуск програмного продукту | 1.Відкриється вікно з програмою |
| 2.Встановлення повзунка «Кількість елементів» на максимальній позначці, а повзунка «Затримка» на мінімальній позначці | 2.Відображення значень кількості елементів та затримки, повзунки знаходяться в вибраних положенях |

Продовження табл. 5.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер  Test  case | Номер  відповідний  вимогам ТЗ | Test case | Опис (сценарій) | Очікуваний результат |
|  |  |  | 3.Запустити виконання програми | 3.Програма виконується |
| ТС-7 | ТЗ-1.3.2 | Перевірити роботу програми при виборі мінімального значення елементів та максимального значення затримки. | 1.Запуск програмного продукту | 1.Відкриється вікно з програмою |
| 2.Встановлення повзунка «Кількість елементів» на мінімальній позначці, а повзунка «Затримка» на максимальній позначці | 2.Відображення значень кількості елементів та затримки, повзунки знаходяться в вибраних положенях |
| 3.Запустити виконання програми | 3.Програма виконується |
| ТС-8 | ТЗ-1.3.2 | Перевірити роботу програми при виборі мінімального значення елементів та мінімального значення затримки. | 1.Запуск програмного продукту | 1.Відкриється вікно з програмою |
| 2.Встановлення повзунка «Кількість елементів» та «Затримка» на мінімальних позначках | 2.Відображення значень кількості елементів та затримки, повзунки знаходяться в вибраних положенях |
| 3.Запустити виконання програми | 3.Програма виконується |

Продовження табл. 5.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер  Test  case | Номер  відповідний  вимогам ТЗ | Test case | Опис (сценарій) | Очікуваний результат |
| ТС-9 | ТЗ-1.3.1 | Перевірити роботу кнопки «Генерувати» після попереднього сортування. | 1.Запуск програмного продукту | 1.Відкриється вікно з програмою |
| 2.Натисути на кнопку «Генерувати» після закінчення попереднього сортування | 2.Відображується  згенерований масив |
| 3.Запустити виконання програми | 3.Програма виконується |
| ТС-10 | ТЗ-1.3.1 | Перевірити стан кнопки «Генерувати» під час роботи програми. | 1.Запуск програмного продукту | 1.Відкриється вікно з програмою |
| 2.Спроба натиснути кнопку «Генерувати» під час роботи програми. | 2.Кнопка «Генерувати» є неактивною |
| 3.Запустити виконання програми | 3.Програма продовжує  виконується |
| ТС-11 | ТЗ-1.3.3 | Перевірити стан кнопки «Сортувати» під час роботи програми. | 1.Запуск програмного продукту | 1.Відкриється вікно з програмою |
| 2.Спроба натиснути кнопку «Сортувати» під час роботи програми. | 2.Кнопка «Сортувати» є неактивною |
| 3.Запустити виконання програми | 3.Програма виконується |

Продовження табл. 5.1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Номер  Test  case | | Номер  відповідний  вимогам ТЗ | Test case | | Опис (сценарій) | Очікуваний результат |
| ТС-12 | | ТЗ-1.3.2 | Перевірити стан повзунка «Кількість елементів» під час роботи програми. | | 1.Запуск програмного продукту | 1.Відкриється вікно з програмою |
| 2.Спроба змінити місцезнаходження позначки «Кількість елементів» під час роботи програми. | 2.Повзунок «Кількість елементів» є неактивним |
| 3.Запустити виконання програми | 3.Програма виконується |
| ТС-13 | ТЗ-1.3.1 | | Перевірити стан повзунка «Затримка» під час роботи програми | 1.Запуск програмного продукту | | 1.Відкриється вікно з програмою |
| 2.Спроба змінити місцезнаходження позначки «Затримка» під час роботи програми. | | 2.Повзунок «Затримка» є неактивним |
| 3.Запустити виконання програми | | 3.Програма виконується |

Продовження табл. 5.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер  Test  case | Номер  відповідний  вимогам ТЗ | Test case | Опис (сценарій) | Очікуваний результат |
| ТС-14 | ТЗ-1.3.2 | Перевірити роботу програми при натиснені кнопки «Сортувати» перш ніж написнути кнопку «Генерувати». | 1.Запуск програмного продукту | 1.Відкриється вікно з програмою |
| 2.Спроба натиснути кнопку «Сортувати» перш ніж натиснути кнопку «Генерувати» | 2.За замовчуванням уже є згенерований масив, який буде сортуватись |
| 3.Запустити виконання програми | 3.Програма виконується |

# 6 ОПИС ЗАСТОСУВАННЯ

При першому запуску програми користувач бачить перед собою основне вікно програми. Це вікно можна побачити на рисунку 6.1:

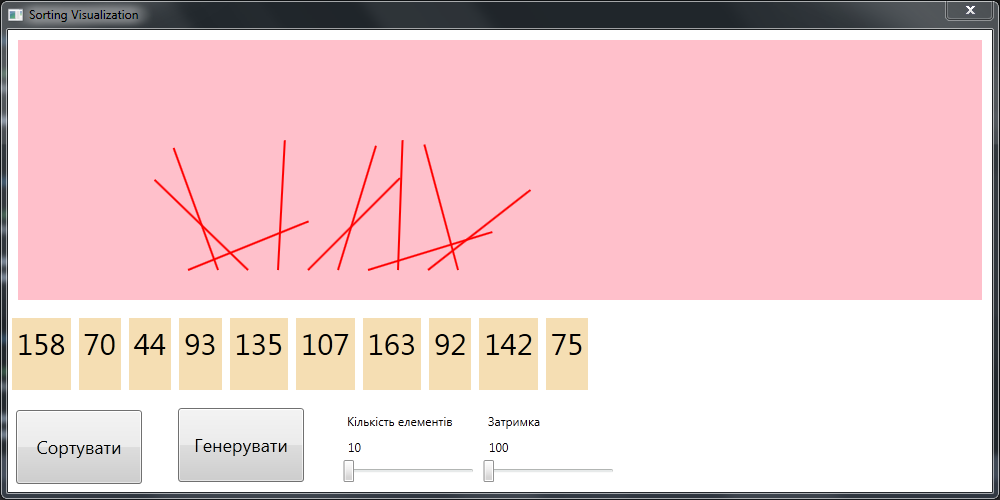


Рисунок 6.1 – Вікно при запуску програми

Це вікно – відправна точка для всіх дій, які користувач може зробити з системою. Вікно розділене на дві частини – область візуалізації (зверху) і область управління (знизу). Область візуалізації використовується для того, щоб візуалізувати роботу системи. Область управління використовується для ініціації будь-яких змін в системі.

Спочатку система надає собою систему з даними по замовчуванням.

Щоб почати генерацію масиву необхідно натиснути кнопку «Генерувати». Після виконання цієї дії ми отримаєм згенерований масив, який буде знаходитись на середині форми, як на рисунку 6.2.

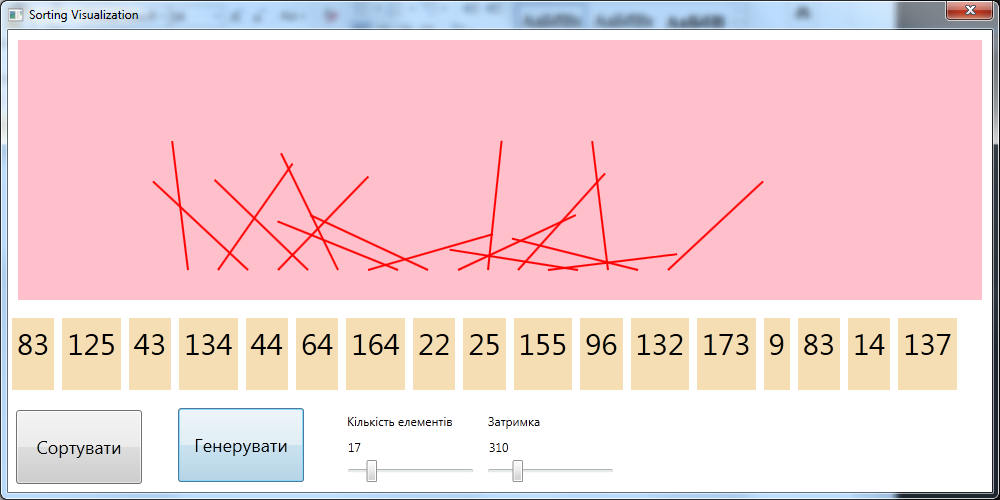


Рисунок 6.2 – Вікно програми з обранимим даними

Для того, щоб почати сортування та візуалізацію потрібно натиснути кнопку «Сортування».

В процесі сортування вікно програми буде виглядати так, як на рисунку 6.3

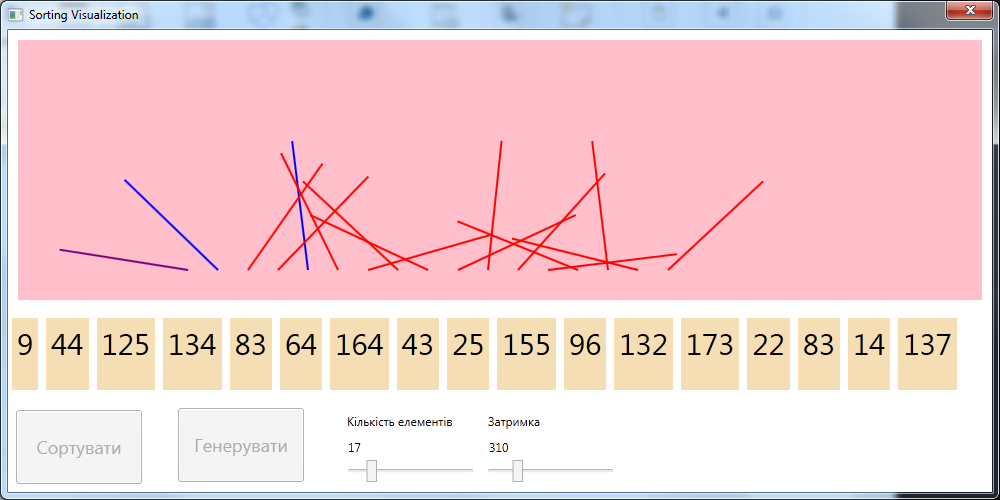


Рисунок 6.3 – Вікно прогрми під час сортування

Фіолетовим кольором зоображені елементи, які являються відсортованими. Синім кольором – елементи, які упорядковуються в даний момент, а червоним – елементи, які ще не були розглянуті.

В результаті ми отримємо відсортований масив та візуальне представлення алгоритму бульбашкового сортування, як на рисунку 6.4:

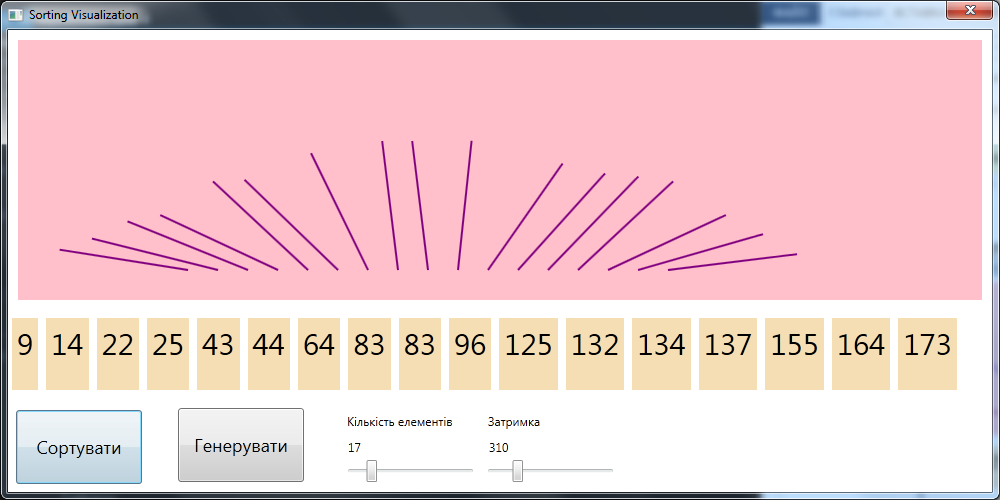


Рисунок 6.4 – Вікно програми з упорядкованим масивом

## ВИСНОВКИ

Метою даної роботи було розглянути систему візуалізації алгоритму бульбашкового сортування за допомогою бібліотеки OpenGL.

Нами було проведено аналіз вимог до функціональності даної системи, обмежень, що накладаються на систему і вхідні параметри. З метою поділу функціональності і сутностей був проведений об'єктний аналіз, наслідком якого стало поділ системи на сукупність базових елементарних класів, які виконують кожен свою елементарну задачу. Була збудована діаграма класів і діаграма модулів для спрощення та систематизації процесу розробки.

У відповідності з проектними документами, була написана програма «Візуалізація алгоритму бульбашкового сортування за допомогою бібліотеки OpenGL». Вона дозволяє сортувати масив згенерованих даних та відобразити сортування за домопомогою графічної бібліотеки OpenGL.

Для більш детального ознайомлення з теоретичною частиною курсового проекту було розглянуто немало різних джерел, які допомогли більш досконально розлянути такі теми: «Діаграми UML», «Бізнес моделювання», «Моделювання систем» і т.д.

Під час роботи над програмою були розглянуті основні етапи тестування.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Советов Б.Я., Яковлев С.А. Моделирование систем. Санкт-Петербург : Высшая школа, 2001.

2. Вендров А.М., Проектирование программного обеспечения экономичес­ких информационных систем. Москва : Финансы и статистика, 2006.

3. СтудопедіЯ. Ваша школопедія. https://studopedia.com.ua/1\_332872\_stsenarii-ta-trasi-podiy.html.

4. Вікіпедія. Діаграма станів (UML). []https://en.wikipedia.org/wiki/UML\_state\_machine#Basic\_UML\_state\_diagrams

5. Microsoft. Випадковий клас. https://docs.microsoft.com/en‑us/dotnet/api/system.random?view=netframework-4.8.

# ДОДАТОК А

***MainWindow.xmal.cs***

using System;

using System.Windows;

using sorting\_visualization.Pattern;

namespace sorting\_visualization

{

delegate void OnOffbuttons(bool tf);

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

Closed += EndProgram;

client.InitializeRandom((int)InputN.Value, (int)InputDelay.Value,Dispatcher,OutputTextBox,DrawingCanvas);

}

private void EndProgram(object sender, EventArgs e)

{

client.CloseWorking();

Close();

}

private void RandomButton\_Click(object sender, RoutedEventArgs e)

{

client.InitializeRandom((int)InputN.Value, (int)InputDelay.Value, Dispatcher, OutputTextBox, DrawingCanvas);

}

private void InputN\_ValueChanged(object sender, RoutedPropertyChangedEventArgs<double> e)

{

Quantity.Content = ((int)InputN.Value).ToString();

}

private void InputDelay\_ValueChanged(object sender, RoutedPropertyChangedEventArgs<double> e)

{

Delay.Content = ((int)InputDelay.Value).ToString();

}

private void ButtonsOnOff(bool tf)

{

RunButton.IsEnabled = tf;

RandomButton.IsEnabled = tf;

InputN.IsEnabled = tf;

InputDelay.IsEnabled = tf;

}

private void RunSortingButton\_Click(object sender, RoutedEventArgs e)

{

OnOffbuttons buttons = ButtonsOnOff;

client.Start(buttons);

}

***Sorting.cs***

using System;

using System.Threading;

using System.Windows.Controls;

using System.Windows.Shapes;

using sorting\_visualization.Pattern.Bridge.Visualization;

namespace sorting\_visualization.Pattern.Bridge

{

static class sorting

{ static public int[] ArrayGeneration(int N)

{ //Генерирование массива значений от 5 до 175

Random random = new Random();

int[] array = new int[N];

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

array[i] = random.Next(5, 175);

return array;

} static public int[] BulbSort(int[] array, Line[] lines,Label[]labels)

{ //сортировка пузырьком

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

{

for (int j = i + 1; j < array.Length; j++)

{

color.SetBlue(i, j, lines); //установка оранжевого цвета

if (array[j] < array[i])

{

int temp = array[j];

array[j] = array[i];

array[i] = temp;

visualization.SwapLines(i, j, lines, array); //изменение положения палочек visualization.PrintArray(array, labels); //печать массива }

Thread.Sleep(bridge.d); //задержка

color.SetRed(i, j, lines); //установка красного цвета

}

color.SetPurple(i, lines);//установка зеленого цвета

}

return array; } }}

***Color.cs***

using System.Threading;

using System.Windows.Media;

using System.Windows.Shapes;

using System.Windows.Threading;

namespace sorting\_visualization.Pattern.Bridge.Visualization

{

static class color

{

static public void SetPurple(int index1, Line[] lines)

{ //изменение цвета палочки на фиолетовый, которые на своём месте

bridge.dispatcher.BeginInvoke(DispatcherPriority.Normal, (ThreadStart)delegate ()

{

lines[index1].Stroke = Brushes.Purple;

});

}

static public void SetBlue(int index1, int index2, Line[] lines)

{

//изменение цвета палочки на синий, которые проверяются

bridge.dispatcher.BeginInvoke(DispatcherPriority.Normal, (ThreadStart)delegate ()

{

lines[index1].Stroke = Brushes.Blue;

lines[index2].Stroke = Brushes.Blue;

});

}

static public void SetRed(int index1, int index2, Line[] lines)

{ //изменение цвета палочки на красный, которые ещё не на своих местах

bridge.dispatcher.BeginInvoke(DispatcherPriority.Normal, (ThreadStart)delegate ()

{ lines[index1].Stroke = Brushes.Red;

lines[index2].Stroke = Brushes.Red;

});

}

}

}

***Visualization.cs***

using System;

using System.Windows.Media;

using System.Windows.Shapes;

using System.Windows.Threading;

using System.Threading;

using System.Windows.Controls;

using System.Windows;

namespace sorting\_visualization.Pattern.Bridge.Visualization

{

static class visualization

{

static private readonly double r = 130;//длина палочки

static private readonly double a = 140, b = 230;//отступы от краёв формы для построения палочек

static public Label[] InitializeLabel(int[] array)

{

//создание ярлыка с нужными характеристиками

Label[] arrayLabel = new Label[array.Length];

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

{

Label label = new Label();

label.Background = Brushes.Wheat;

label.FontSize = 30;

label.Content = array[i];

label.BorderThickness = new Thickness(4);

arrayLabel[i] = label;

}

return arrayLabel;

}

static public Line[] InitializeLines(int[] array)

{ //создание палочки с нужными характеристиками

Line[] arrayLine = new Line[array.Length];

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

{

Line line = new Line();

line.X1 = a + ((i + 1) \* 30);

line.Y1 = b;

line.X2 = -Math.Cos(((double)array[i] / 180) \* Math.PI) \* r + line.X1;

line.Y2 = -Math.Sin(((double)array[i] / 180) \* Math.PI) \* r + line.Y1;

line.StrokeThickness = 2;

line.Stroke = Brushes.Red;

arrayLine[i] = line;

}

return arrayLine;

}

static public void SwapLines(int index1, int index2 , Line[] lines , int[] array)

{

//изменение положения двух палочек

bridge.dispatcher.BeginInvoke(DispatcherPriority.Normal, (ThreadStart)delegate ()

{

double Onex1 = lines[index1].X1;

double Oney1 = lines[index1].Y1;

double Twox1 = lines[index2].X1;

double Twoy1 = lines[index2].Y1;

lines[index2].X1 = Onex1;

lines[index2].Y1 = Oney1;

lines[index2].X2 = -Math.Cos(((double)array[index2] / 180) \* Math.PI) \* r + lines[index2].X1;

lines[index2].Y2 = -Math.Sin(((double)array[index2] / 180) \* Math.PI) \* r + lines[index2].Y1;

lines[index1].X1 = Twox1;

lines[index1].Y1 = Twoy1;

lines[index1].X2 = -Math.Cos(((double)array[index1] / 180) \* Math.PI) \* r + lines[index1].X1;

lines[index1].Y2 = -Math.Sin(((double)array[index1] / 180) \* Math.PI) \* r + lines[index1].Y1;

});

Line temp = lines[index1];

lines[index1] = lines[index2];

lines[index2] = temp;

}

static public void PrintArray(int[] array, Label[] labels)

{

//печать массива значений

bridge.dispatcher.BeginInvoke(DispatcherPriority.Normal, (ThreadStart)delegate ()

{

bridge.PanelForLines.Children.Clear();

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

labels[i].Content = array[i].ToString();

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

bridge.PanelForLines.Children.Add(labels[i]);

}); } }}

***Bridge.cs***

using System.Threading;

using System.Windows.Controls;

using System.Windows.Shapes;

using System.Windows.Threading;

using sorting\_visualization.Pattern.Bridge.Visualization;

namespace sorting\_visualization.Pattern.Bridge

{

static class bridge

{

static public int n = 20;//минимальное количество значение

static public int d = 100;//минимальная задержка

static public int[] numbers;//массив значений

static public Line[] lines;//массив линий

static public Label[] labels; //массив ярлыков

static public Thread SortingThread;//экземпляр класса Thread для отдельного потока

static public Dispatcher dispatcher;

static public StackPanel PanelForLines;

static public Grid GridForNumbers;

static public void GenerationArray()

{

numbers = sorting.ArrayGeneration(n);

}

static public void LinesInitialize()

{

lines = visualization.InitializeLines(numbers);

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

GridForNumbers.Children.Add(lines[i]);

}

static public void LabelsInitialize()

{

labels = visualization.InitializeLabel(numbers);

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

PanelForLines.Children.Add(labels[i]);

}

static public void StartSort()

{

numbers = sorting.BulbSort(numbers, lines, labels);

}

}

}

***Client.cs***

using System.Threading;

using System.Windows.Threading;

using System.Windows.Controls;

using sorting\_visualization.Pattern.Bridge;

namespace sorting\_visualization.Pattern

{

static class client

{

static public void InitializeRandom(int N, int D, Dispatcher dispatcher , StackPanel stackpanel, Grid grid)

{

//инициализация палочек и значений

bridge.n = N;

bridge.d = D;

bridge.dispatcher = dispatcher;

bridge.PanelForLines = stackpanel;

bridge.GridForNumbers = grid;

bridge.GenerationArray();

bridge.GridForNumbers.Children.Clear();

bridge.LinesInitialize();

bridge.PanelForLines.Children.Clear();

bridge.LabelsInitialize();

}

static public void CloseWorking()

{

if (bridge.SortingThread != null)

bridge.SortingThread.Abort();

}

static public void Start(OnOffbuttons buttons)

{

buttons(false);

bridge.SortingThread = new Thread(delegate ()

{

bridge.StartSort();

bridge.dispatcher.BeginInvoke(DispatcherPriority.Normal, (ThreadStart)delegate ()

{

buttons(true);

});

});

bridge.SortingThread.Start();

}

}

}