Міністерство освіти і науки України
Український державний університет науки і технологій

Факультет «Комп'ютерні технології і системи»

Кафедра «Комп'ютерні інформаційні технології»

Пояснювальна записка
до кваліфікаційної роботи бакалавра

на тему: «Розробка системи аналізу часових характеристик використання динамічної пам'яті в ОС Windows»
за освітньою програмою: «Інженерія програмного забезпечення»
зі спеціальності: «121 Інженерія програмного забезпечення»
Виконав: студент групи «ПЗ1811»

_____ /Андрій МОРЕНКО/
(підпис студента) (Ім'я ПРІЗВИЩЕ)

Керівник: _____ /доц. Вадим АНДРЮЩЕНКО/
(підпис) (посада, Ім'я ПРІЗВИЩЕ)

Нормоконтролер: _____ /доц. Олена КУРОП'ЯТНИК/
(підпис) (посада, Ім'я ПРІЗВИЩЕ)

Консультанти:

Засвідчую, що у цій роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Дніпро – 2022 рік

Ministry of Education and Science of Ukraine
Ukrainian State University of Science and Technologies

Faculty «Computer technologies and systems»

Department «Computer information technology»

Explanatory Note
to Bachelor's Thesis

on the topic: «Development of a time analysis system characteristics of the use of dynamic memory in Windows»
according to educational curriculum «Software engineering»
in the Speciality: «121 Software engineering»

Done by the student of the group PZ1811: /Andrii MORENKO/

Scientific Supervisor: /Vadym ANDRIUSHCHENKO/

Normative controller: /Olena KUROPIATNYK/

Dnipro – 2022

Міністерство освіти і науки України
Український державний університет науки і технологій

Факультет:«Комп'ютерні технології і системи»
Кафедра: «Комп'ютерні інформаційні технології»
Рівень вищої освіти: бакалавр
Освітня програма: «Інженерія програмного забезпечення»
Спеціальність: «121 Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри КІТ
_____ /Вадим ГОРЯЧКІН/
(підпис)

Дата _____

З А В Д А Н Н Я

на кваліфікаційну роботу бакалавра
студенту Моренку Андрію Олександровичу
1. Тема роботи: «Розробка системи аналізу часових характеристик використання динамічної пам'яті в ОС Windows»
Керівник роботи: Андрющенко Вадим Олександрович, доцент
затверджені наказом № 77 ст від 08.12.2021
2. Строк подання студентом роботи: 11.06.2022 р.
3. Вихідні дані до роботи: _____

4. Зміст пояснювальної записки (перелік питань, які потрібно опрацювати): вступ, збір вимог до програмного забезпечення, огляд літератури, проектування, тестування та налагодження.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): презентація, відео роботи програми

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Постановка задачі	31.01.2022 – 05.01.2022	
2	Огляд літератури та аналіз аналогів	16.01.2022 – 24.01.2022	
3	Розробка структур вхідних і вихіних даних	25.01.2022 – 02.02.2022	
4	Визначення вимог до програми. Вибір та обгрунтування мови програмування	03.02.2022 – 10.02.2022	
5	Узгодження та затвердження ТЗ	11.02.2022 – 18.02.2022	
6	Розробка та програмування логіки програми	19.02.2022 – 01.03.2022	
7	Розробка і реалізація інтерфейсу користувача	02.03.2022 – 20.03.2022	
8	Відлагодження програми	21.03.2022 – 24.03.2022	
9	Розробка, узгодження та затвердження програмної документації	25.03.2022 – 03.04.2022	
10	Подання кваліфікаційної роботи до кафедри	04.04.2022– 12.06.2022	
11	Захист кваліфікаційної роботи на засіданні Екзаменаційної комісії	21.06.2022	

Студент _____ (підпис) Андрій МОРЕНКО (Ім'я ПРІЗВИЩЕ)

Керівник роботи _____ (підпис) доц. Вадим АНДРЮЩЕНКО (Ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка складається з 4 розділів:

– вступ – у даному розділі описується сутність розробки, її актуальність. Складається з 1 сторінки;

– збір вимог до програмного забезпечення – у цьому розділі описуються аналоги програми та література по даній предметній області, а також проводиться опитування зацікавлених сторін для формування найбільш повних вимог до програмного забезпечення. Складається з 3 сторінок;

– проектування усієї системи – у цьому розділі проведений огляд вхідних і вихідних даних, формалізація задачі, розробка фізичного проекту, проектування інтерфейсу користувача, аналіз проекту, вибір мови програмування. Складається з 21 сторінки;

– тестування та налагодження – включає в себе вибір стратегії тестування, опис тестів методами «чорного» та «білого» ящика. Також аналіз помилок їх вплив на систему та вирішення проблеми. Складається з 9 сторінок;

– висновки. Складається з 1 сторінки;

– список літератури включає в себе бібліографічний список використаної літератури. Складає 1 сторінку;

– додатки – містить технічне завдання

Кількість таблиць: 4 штуки.

Кількість рисунків: 22 штуки.

Ключові слова: Heap, Stack, динамічна пам’ять, ООП, таблиці, діаграми, динамічна пам’ять.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАК, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

QT – крос-платформовий інструментарій розробки програмного забезпечення (ПЗ) мовою програмування C++. Дозволяє запускати написане за його допомогою ПЗ на більшості сучасних операційних систем (ОС), просто компілюючи текст програми для кожної операційної системи без зміни сирцевого коду.

Динамічна оперативна пам'ять або DRAM (Dynamic Random Access Memory) — один із видів комп'ютерної пам'яті із довільним доступом (RAM), найчастіше використовується як ОЗП сучасних комп'ютерів.

CMake – крос-платформовий відкритий генератор сценаріїв складання. CMake не займається безпосередньо складанням, а лише генерує файли управління складанням з файлів CMakeLists.txt.

ВСТУП

Суть розробки проекту полягає у створенні програми для аналізу використання динамічної пам’яті в ОС Windows. Програма передбачає у собі наявність певних інструментів для аналізу використання динамічної пам’яті у режимі реального часу. Ці інструменти допомагають краще розуміти як використовується та розподіляється динамічна пам’ять.

За допомогою програмного забезпечення, що розроблюється, можна буде бачити усі процеси які використовують динамічну пам’ять. Також у даній програмі користувач матиме можливість аналізувати показники у вигляді таблиці та графіку. У програмі буде можливо відстежувати розподілення та перерозподіл пам’яті. Що дозволяє користувачу аналізувати які процеси використовують динамічну пам’ять.

Актуальність роботи: на сьогоднішній день дана робота є досить актуальною, оскільки розумне використання пам’яті дозволяє оптимізувати програми які розробляються для ОС Windows. Це необхідно для того, щоб уникнути перенавантаження операційної системи. Велика кількість розробників та компаній зараз використовує різні програми для аналізу використання динамічної пам’яті. Тому, розробка даного проекту допоможе користувачу краще розуміти, що відбувається з динамічною пам’яттю в режимі реального часу. Також під час розробки даного проекту, було розглянуто переваги та недоліки існуючих аналогів та проведено опитування зацікавлених сторін, завдяки чому було покращено кінцевий продукт.

РОЗДІЛ 1 ЗБІР ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Огляд аналогів

В ході розробки програми дипломної роботи, було розглянуто аналоги програм які демонструють роботу с динамічною пам’яттю, визначено їх переваги та недоліки, покращення власної програми завдяки висновкам, зроблених під час огляду аналогів. Основна частина – це десктопні додатки. Частина функціоналу безкоштовна. Проте, для користування повним функціоналом потрібно оформити платну підписку. До уваги обрано найпопулярніші продукти ринку.

Intel VTune Amplifier та AMD Performance Analyzer ці інструменти можуть виконувати типові функції профілювання пам'яті. Два інструменти, пропонуються компаніями AMD та Intel на платній основі. Вони пропонують набагато глибший аналіз, що далеко виходить за межі стандартного профілювання, що дозволяє дізнатися, які методи працюють найдовше. Можна отримувати інформацію від апаратних лічильників, вбудованих в обладнання (процесор) і повідомляють про його внутрішню поведінку: використання кеша і пам'яті, простоювання конвеєра та багато іншого.

Переваги: користувачу надається можливість переглянути документацію на сайті та пройти навчання. Досить значна частина контенту – безкоштовна. Дуже великий функціонал, завдяки чому користувач має багато можливостей. Також на сайті розробника є вебінари на яких користувач міг дізнатись багато корисної інформації стосовно програмного продукту.

Недоліки: для використання подібних інструментів потрібно розбиратись: в обладнанні, середовищі виконання і навіть мові асемблера. Також важко розібратися з усіма можливостями без перегляду відео-уроків та вебінарів. Цей продукт більше підходить для розробників програмних продуктів специфіка яких передбачає роботу з динамічною пам'яттю. Менше ці програми слугують для постійного використання пересічним

користувачем, оскільки для зручного використання буде необхідно витратити багато часу для освоєння програми.

JetBrains DotMemory загальний функціонал та інтерфейс: Для виконання поглибленого аналізу програма дозволяє застосовувати до даних з використання пам'яті безліч різних критеріїв, що дає можливість переглядати дані з тисяч різних ракурсів, деталізувати їх, виконувати площинні і об'ємні зрізи, а також обертання. Більш складний інтерфейс ніж у попередніх аналогів, він має певні відмінності, проте суть не змінюється. Проте, присутні безкоштовні функції, що недоступні у першому варіанті. На сайті розробника є відеотур який дозволить ознайомитись с інтерфейсом програми. У цій програмі немає «перевантаженості» інтерфейсу, як у попередньому аналогу.

Переваги: зручний інтерфейс, в якому легко розібратися. Більшість функціоналу зрозуміла на інтуїтивному рівні. Присутні підказки, що допомагають розібратися під час використання . Присутня інтеграція з іншими популярними сервісами, такими як: Visual Studio.

Недоліки: основним недоліком цієї програми є велика вартість. Через це не кожен може дозволити собі цей програмний продукт.

1.2 Опитування зацікавлених сторін

Опитування проводилося серед розробників та пересічних користувачів. Інструмент опитування: google-форми. Для зацікавлених сторін було представлено основні питання, які допомогли би під час подальшого проектування програмного забезпечення.

Рисунок 1.1 – Сфера діяльності

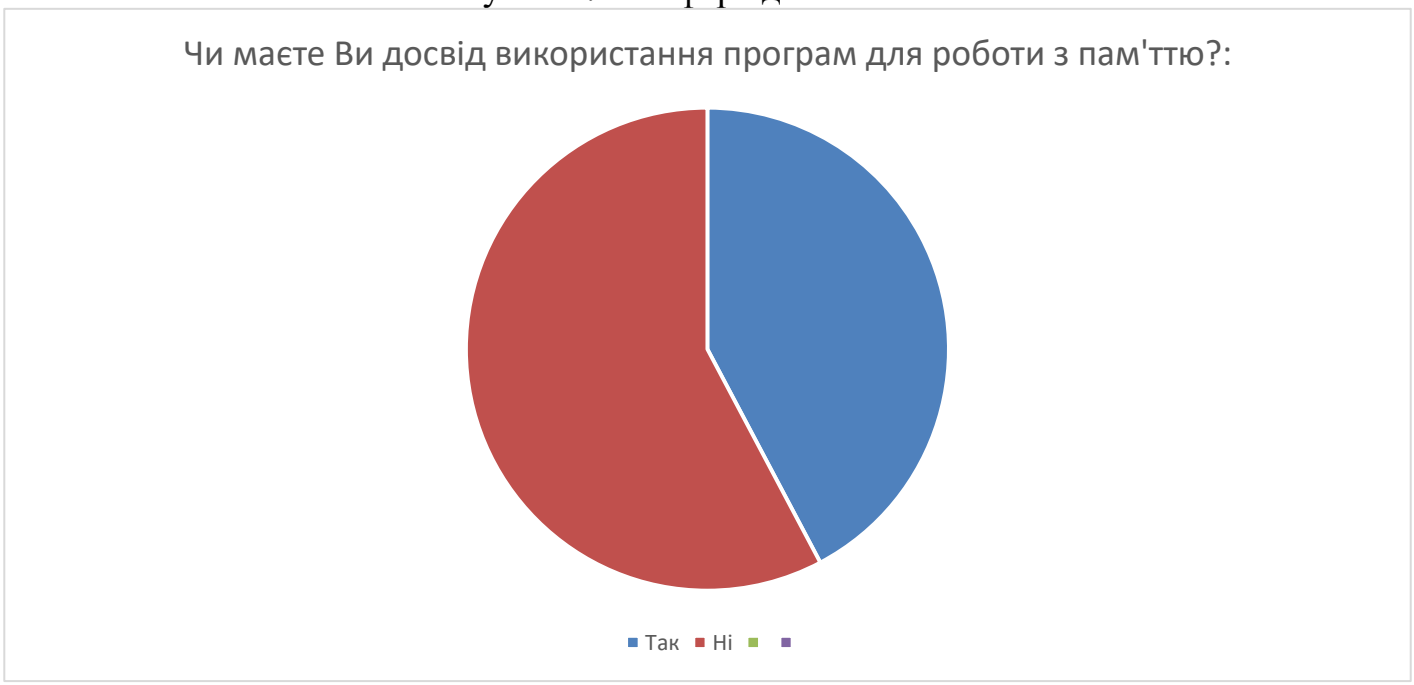

Рисунок 1.2 – Досвід використання аналогів

Рисунок 1.3 – Види показників які будуть реалізовані у програмі

1.3 Постановка задачі

Першою задачею являється визначення загальних переваг та недоліків основних аналогів на ринку, а також необхідно зробити висновки стосовно покращення власного проекту. Після цього провести опитування зацікавлених сторін та підкреслити для себе найголовніші потреби користувачів. На основі потреб користувачів розробити власний додаток.

Висновки до розділу 1

Аналіз та збір вимог під час проектування програмного забезпечення є важливим етапом при розробці. Це допомагає спланувати подальшу роботу. За допомогою аналізу вимог можливо побачити основні недоліки аналогів ринку, що дає можливість виправити ці недоліки у своєму проекті.

Було опитано зацікавлені сторони (програмістів, студентів та пересічних користувачів які не пов’язані з IT). Зібрано статистику основних потреб користувачів, а також зібрано інформацію про досвід використання схожих (аналогічних) програм. Використовуючи отриману інформацію, створено план подальшої роботи.

РОЗДІЛ 2 ОГЛЯД ЛІТЕРАТУРИ

2.1 Область оперативної пам’яті: Stack

Стек – це область оперативної пам'яті, що створюється кожному за потоку. Він працює в порядку LIFO (Last In, First Out), тобто останній доданий у стек шматок пам'яті буде першим у черзі на виведення зі стека. Щоразу, коли функція оголошує нову змінну, вона додається в стек, а коли ця змінна зникає з області видимості (наприклад, коли функція закінчується), вона автоматично видаляється зі стека. Коли стікова змінна звільняється, ця область пам'яті стає доступною для інших стікових змінних. Через таку природу стека управління пам'яттю виявляється дуже логічним і простим для виконання на ЦП; це призводить до високої швидкості, особливо оскільки час циклу оновлення байта стека дуже мало, тобто. цей байт найімовірніше прив'язаний до кешу процесора. Тим не менш, у такої жорсткої форми управління є і недоліки. Розмір стека – це фіксована величина, і перевищення ліміту виділеної на стеку пам'яті призведе до переповнення стека. Розмір задається при створенні потоку, і кожна змінна має максимальний розмір, що залежить від типу даних. Це дозволяє обмежувати розмір деяких змінних (наприклад, цілих), і змушує заздалегідь оголошувати розмір більш складних типів даних (наприклад, масивів), оскільки стек не дозволить їм змінити його. Крім того, змінні, розташовані на стеку, є локальними. У результаті стек дозволяє управляти пам'яттю найбільш ефективним чином - але якщо вам потрібно використовувати динамічні структури даних або глобальні змінні, варто звернути увагу на купу[7].

2.2 Область оперативної пам’яті: Heap

Купа – це сховище пам'яті, також розташоване в ОЗУ, яке дозволяє динамічне виділення пам'яті і не працює за принципом стека: це просто склад для ваших змінних. Коли ви виділяєте в купі ділянку пам'яті для зберігання змінної, до неї можна звернутися не тільки в потоці, але й у всьому додатку. Саме так визначаються глобальні змінні. Після завершення програми всі

виділені ділянки пам'яті звільняються. Розмір купи задається при запуску програми, але, на відміну стека, він обмежений лише фізично, і це дозволяє створювати динамічні змінні[7].

2.3 Порівняння областей оперативної пам’яті

У порівнянні зі стеком купа працює повільніше, оскільки змінні розкидані по пам'яті, а не сидять на верхівці стека. Некоректне керування пам'яттю в купі призводить до уповільнення її роботи; тим не менш, це не зменшує її важливості - якщо необхідно потрібно працювати з динамічними або глобальними змінними, потрібно користуватись купою.

2.4 Опис об’єктно-орієнтованого програмування

За визначенням визнаного авторитету у галузі об'єктно-орієнтованих методів розробки програм Граді Буча «об'єктно-орієнтоване програмування (ООП) — це методологія програмування, яка заснована на представленні програми у вигляді сукупності об'єктів, кожен з яких є реалізацією певного класу (типу особливого виду), а класи утворюють ієрархію за принципами успадкованості».

Об'єктно-орієнтований підхід до проектування програмного продукту ґрунтується на:

• виділення класів об'єктів;

• встановлення характерних властивостей об'єктів та методів їх обробки;

• створення ієрархії класів, успадкування властивостей об'єктів та методів їх обробки.

Кожен об'єднує як дані, і програму обробки цих даних і належить до певного класу. За допомогою класу один і той же програмний код можна використовувати для різних об'єктів, що відносяться до нього.

Об'єктний підхід розробки алгоритмів і програм передбачає:

• об'єктно-орієнтований аналіз – аналіз предметної області та виділення об'єктів, визначення властивостей та методів обробки об'єктів, встановлення їх взаємозв'язків;

• об'єктно-орієнтоване проектування – поєднує процес об'єктної декомпозиції та подання з використанням моделей даних проектованої системи на логічному та фізичному рівнях, у статиці та динаміці.

Механізми, що підтримуються ООП, дозволяють моделювати поняття предметної області більш прямим та природним шляхом.

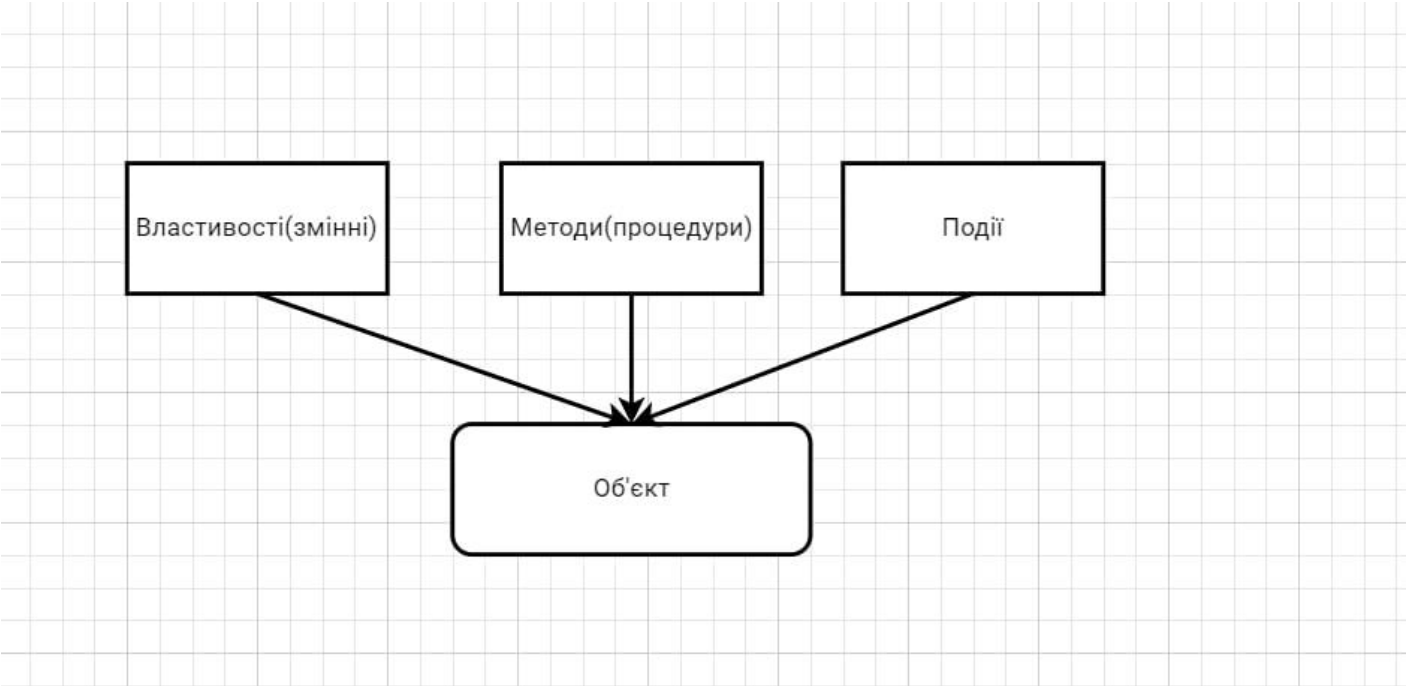

Рисунок 2.1 Графічне представлення об'єкта

Визначальним поняттям ООП є об'єкт – деяка сукупність, що об'єднує властивості, методи та події. Об'єкт представляє конкретну реалізацію класу, що володіє характеристиками стану, поведінки та індивідуальності. В ООП об'єкти – це окремі фрагменти коду та дані, які взаємодіють один з одним та навколишнім світом, моделюючи стан та взаємодію об'єктів реального світу. Об'єкти належать класам (типам) - безлічі об'єктів, пов'язаних певною спільністю структури та поведінки[4].

РОЗДІЛ 3 ПРОЕКТУВАННЯ

3.1 Визначення вхідних і вихідних даних

Програма повинна забезпечити можливість бачити використання динамічної пам’яті в режимі реального часу.

В програмі необхідно реалізувати можливість налаштовувати небхідні для аналізу таблиці та діаграми.

Вхідні данні: показники з використання динамічної пам’яті.

Вихідні дані: побудовані діаграми та таблиці.

Вихідні дані виводяться у вигляді діаграм та таблиць.

У програмі інтерфейс має бути інтуїтивно зрозумілим для простоти використання. Кожен із розділів інтерфейсу користувача присвячений різної візуалізації використання пам'яті.

3.2 Формалізація задачі

Функціональне призначення – програмний продукт являє собою програму для аналізу часових характеристик використання динамічної пам’яті в OS Windows.

Для аналізу буде представлено 2 діаграми та 4 види табличок:Memory timeline,Allocation та Statistics, Memory tag, Heap,Stack.

Існує можливість аналізувати показники в режимі реального часу.

Експлуатаційне призначення – збір та аналіз даних з використання динамічної пам’яті в режимі реального часу.

3.3 Розробка фізичного проекту

Детальніше розглянемо основні процеси, які відбуваються в системі. Для цього використовують діаграми. На рисунку 3.1 показана діаграма

проектованої системи.

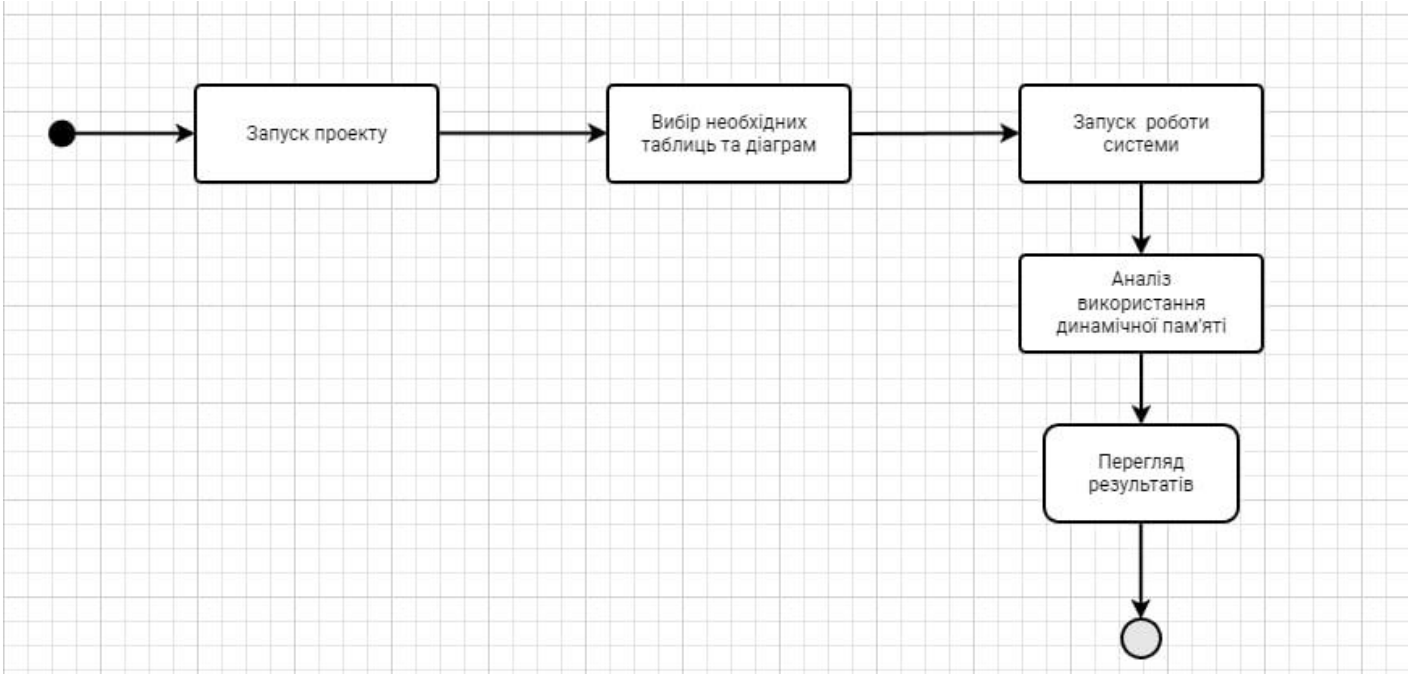

Рисунок 3.1 Діаграма проектованої системи

3.4 Проектування архітектури системи

Необхідність у проектуванні архітектури системи була зумовлена тим, що у програмі реалізована велика кількість функціоналу. Від збирання даних до відображення їх у таблицях та діаграмах. Архітектура програми повинна включає в себе всі програмні модулі і компоненти та їх взаємодія між собою.

Перед проектуванням архітектури було проведено певну підготовку, а саме:

– попередні дослідження;
– формування вимог;
– аналіз вимог;
– користувацькі і системні вимоги;
– специфікація вимог.

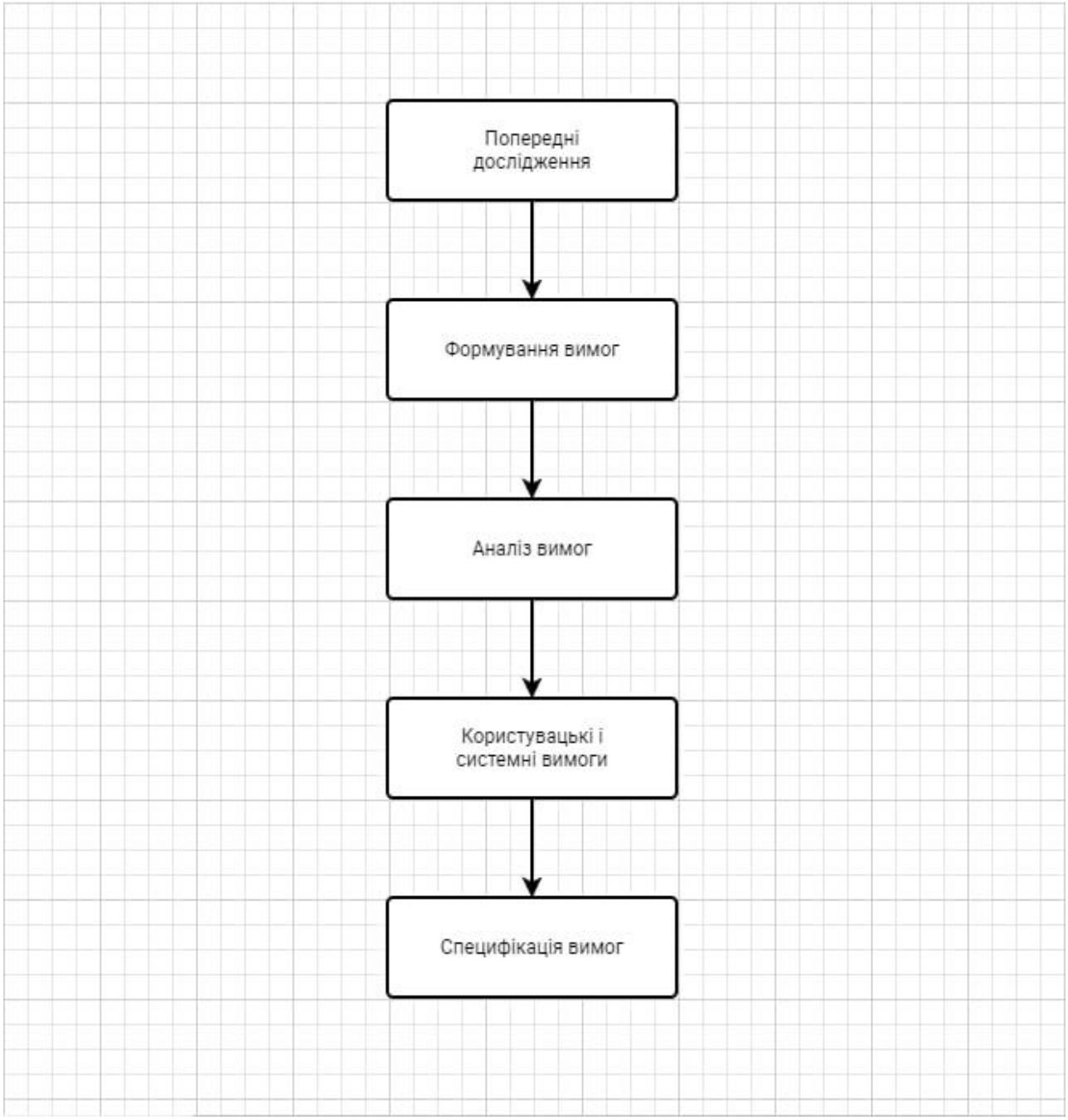

Рисунок 3.2 Дії які було проведено перед проектуванням системи

Для того, щоб у результаті було створено структурований додаток, перед стадією розробки було проаналізовано усі етапи роботи програми які необхідно реалізувати. При цьому кожному етапі виконуються різні дії.

Мета використання такої моделі – створити ефективний, економічно вигідний та якісний програмний продукт.

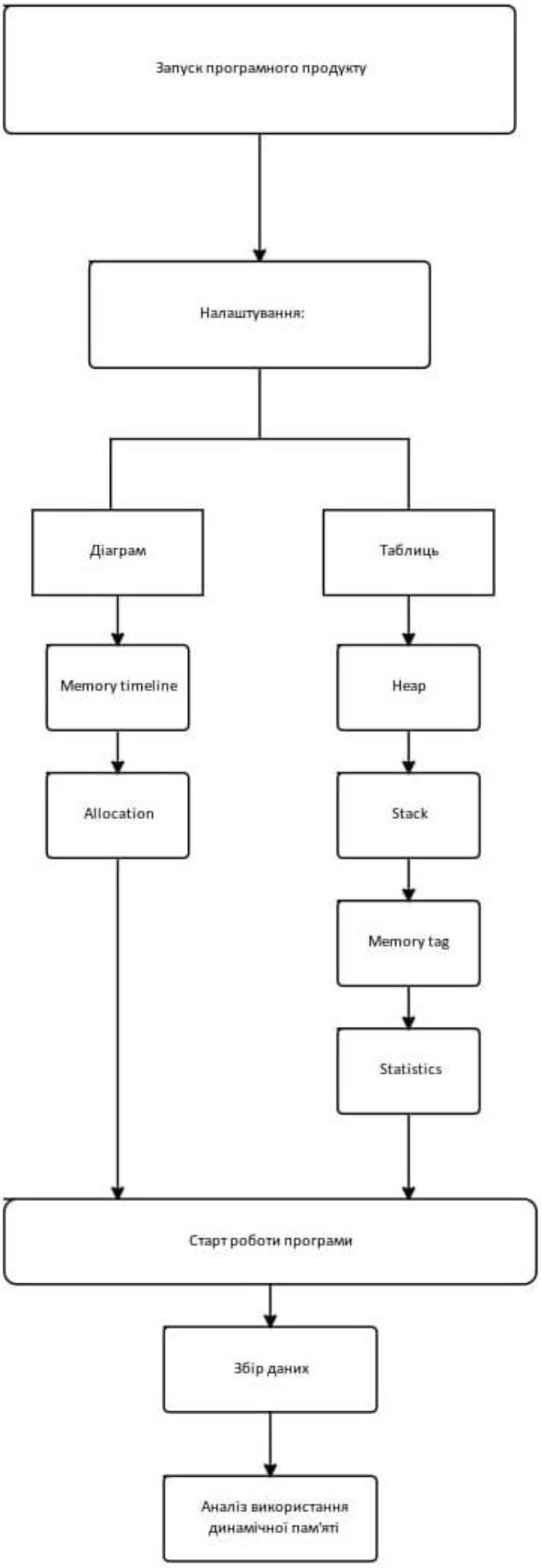

Рисунок 3.3 Прототип програмного забезпечення

Перед проектуванням архітектури усієї системи було визначено, які технології необхідні для реалізації програмного забезпечення. Певні технології були використані для того, щоб проект можливо було переносити між різними IDE.

Для цього використовувалася кросплатформова система автоматизації збирання програмного забезпечення CMake. CMake є генератором сценаріїв складання, і таким чином, робить проект переносимим між різними IDE. Це було необхідно оскільки певні частини проекту були розроблені у Visual Studio 2022, а для реалізації інтерфейсу програмного забезпечення використовувалась Qt Creator.

Оскільки для програмного продукту мовою програмування було обрано C++, а програма повинна працювати на операційній системі Windows. Для цього було використано набір інструментів MinGw (Minimalist GNU for Windows). Він включає в себе компілятор разом з набором вільно розповсюджуваних бібліотек імпорту та заголовних файлів для Windows API.

Також для реалізації цього проекту використовувались (DDL). Це – динамічна бібліотека, в якій програма може зберігати функції та змінні.

Оскільки програмне забезпечення збирає велику кількість даних та має велике навантаження, є логічним використовувати динамічну бібліотеку. Вона містить у собі функції, які завантажуються в ОЗУ тільки у той момент, коли вони реально знадобились. Наприклад функції зі статичних бібліотек прикомпоновуються до модуля заздалегідь, незалежно від того, чи будуть вони викликані, чи ні.

Необхідно також згадати про Win32 API(WinApi). Використовуючи API, вдалось отримати деякі дані роботи потоків та використання динамічної пам’яті.

Для компіляції деяких модулів програми було залучено компілятор «GCC», його використовують для компіляції коду на мові C++ та C.

Загалом це повний список технологій за допомогою яких вдалось створи програмне забезпечення з аналізу використання динамічної пам’яті.

Архітектуру розробленого програмного забезпечення продемонстровано у вигляді розроблених класів та методів та пояснені до них. Завдяки чому можливо створили повну картину того, яку роботу було зроблено і як працює програма.

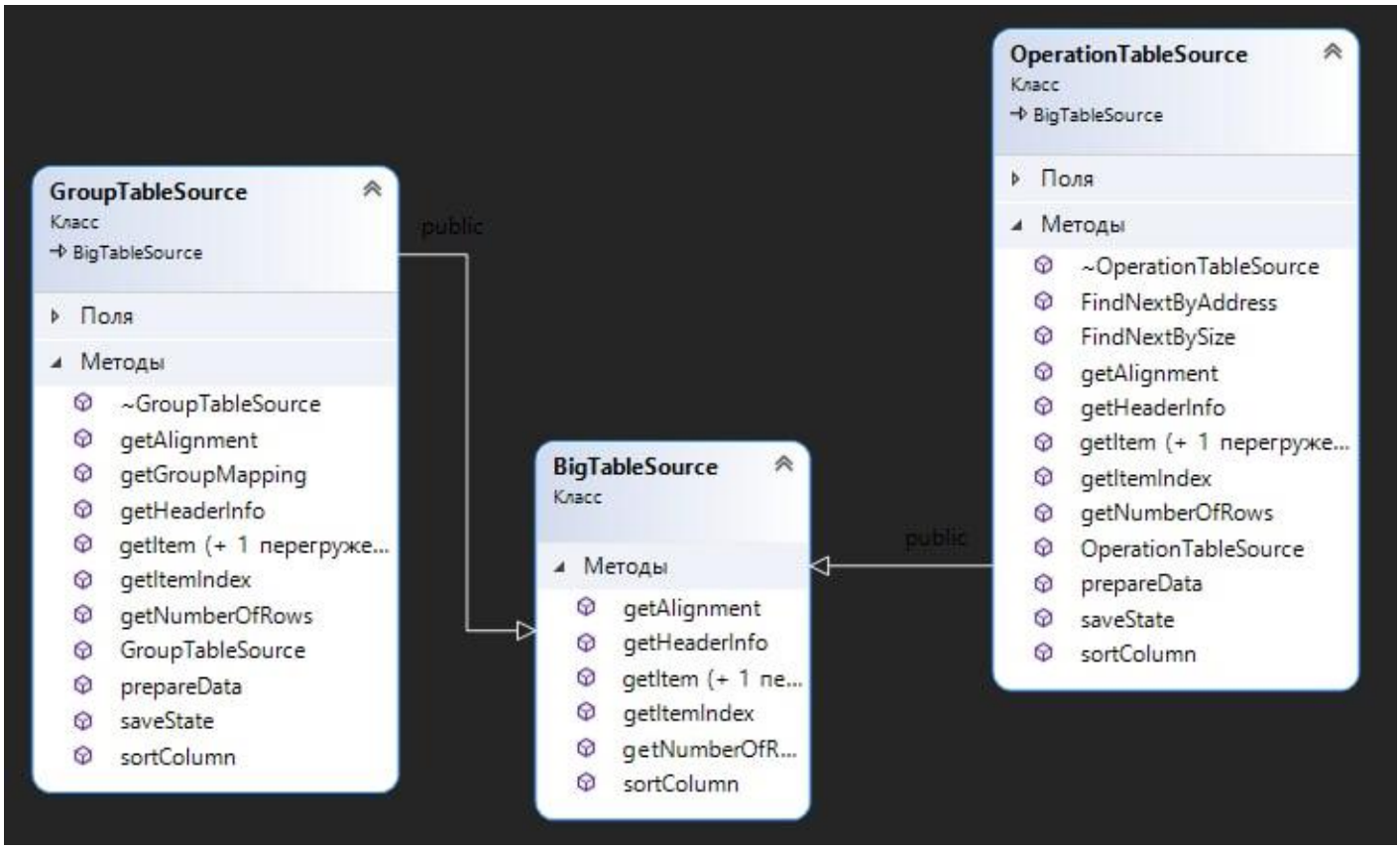

Рисунок 3.4 Класи та їх методи які відповідають за збір та відображення даних в табличках

У цих класах реалізовано збір даних які необхідні для аналізу даних з використання динамічної пам’яті. Також методи у цих класах відповідають за відображення усіх показників у табличках.

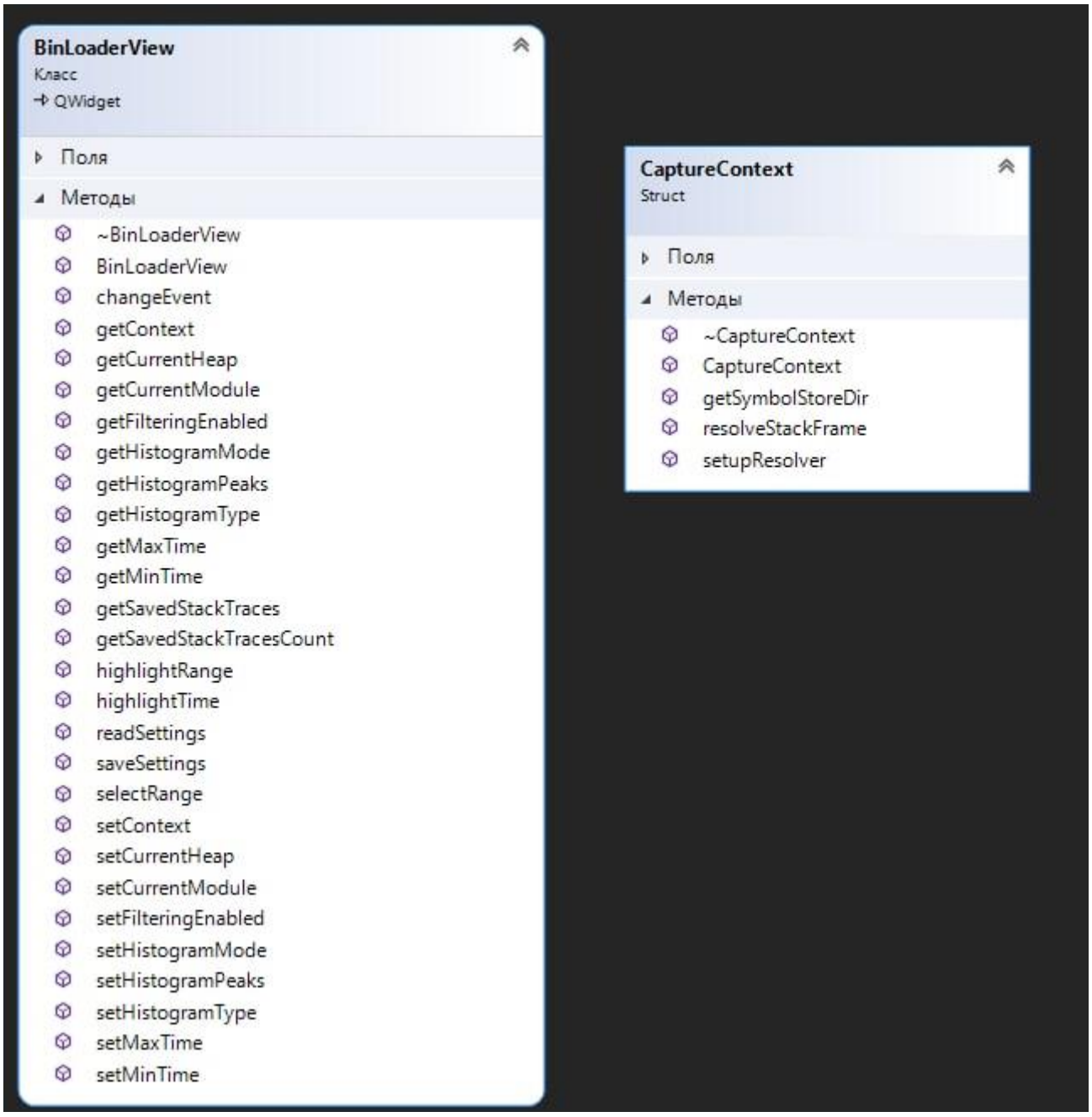

Рисунок 3.5 Класи та їх методи які відповідають за збір та відображення даних в діаграмах

У цих класах продемонстровано роботу з отримання показників пам’яті, скільки часу було витрачено на виділення динамічної пам’яті. Також, за допомогою цих методів відображається інформація в діаграмах.

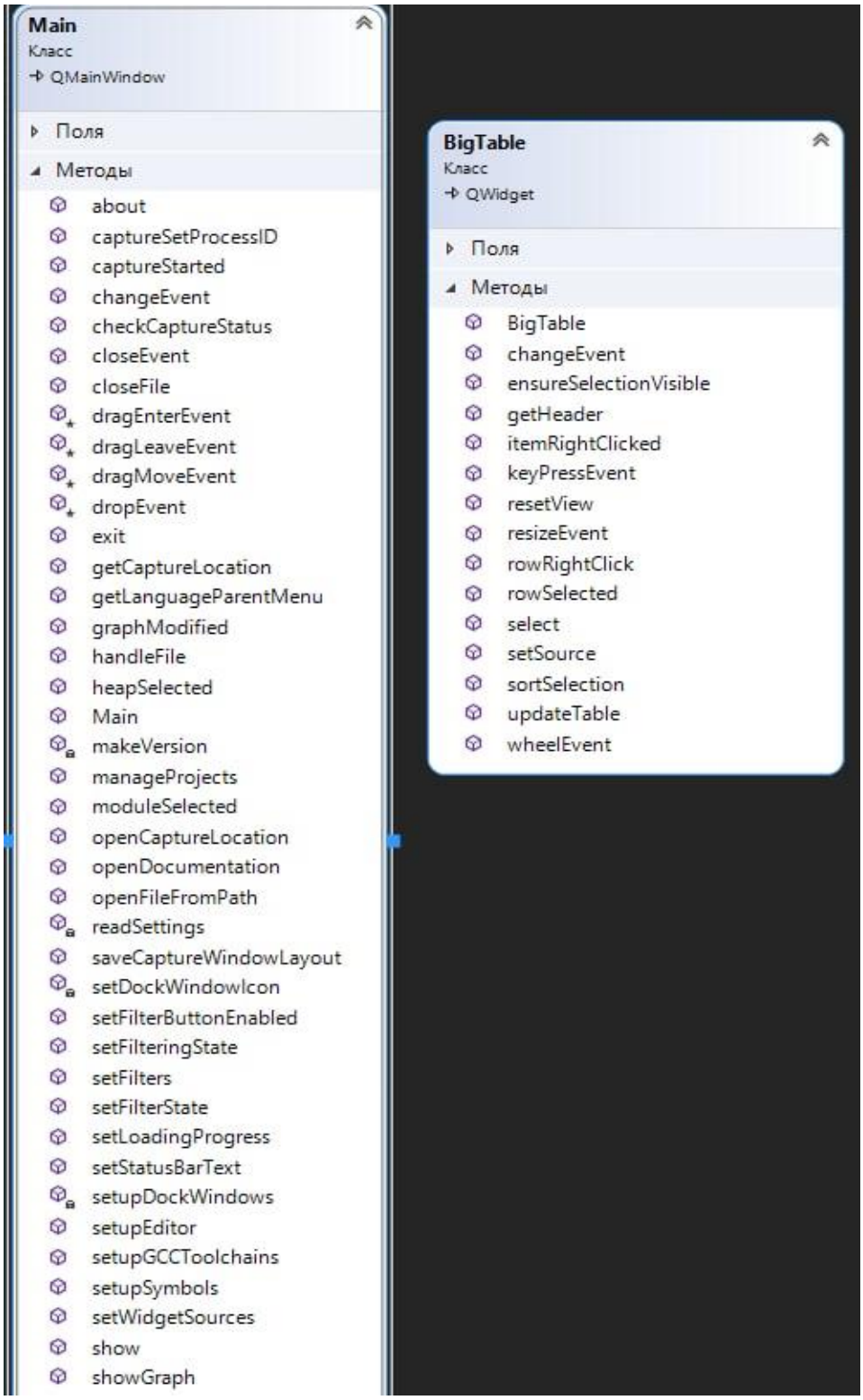

Рисунок 3.6 Головний клас та табличка з даними про використання динамічної пам’яті

Тут зображено методи головного класу, та методі класу який відповідає за відображення даних в табличці.

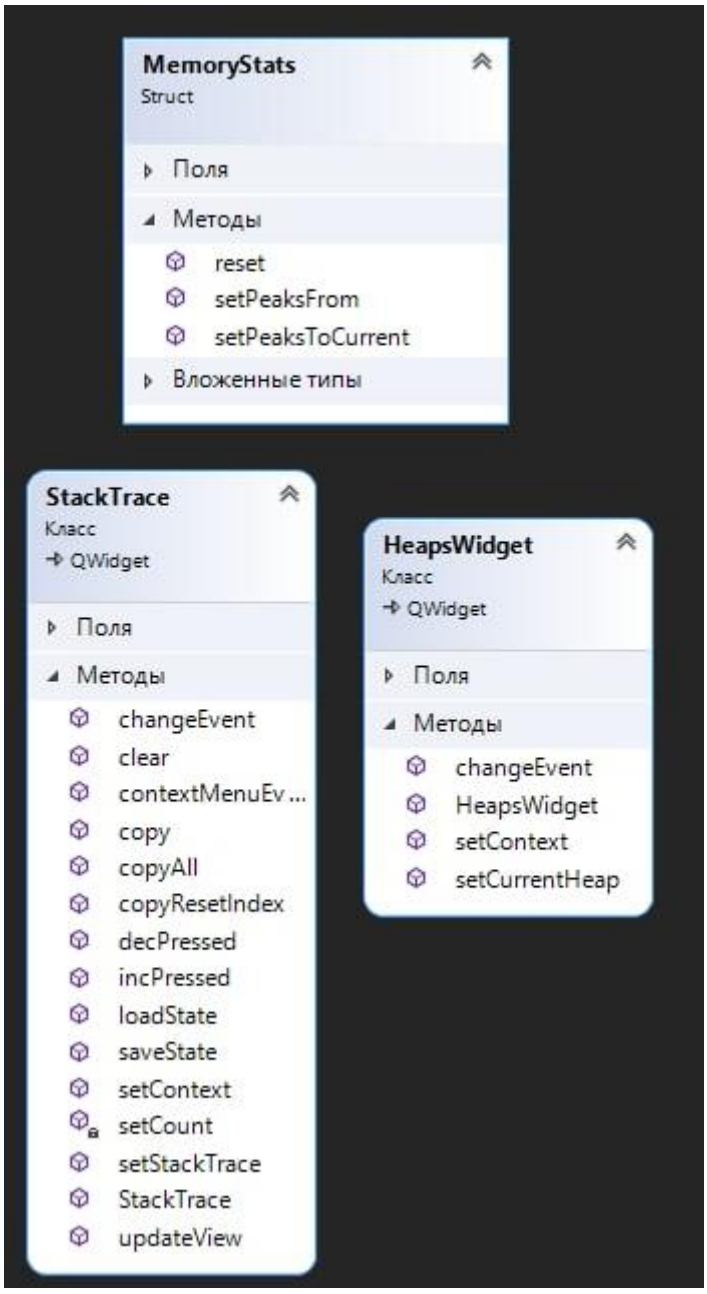

Рисунок 3.7 Основні класи які відповідають за роботу з пам’яттю

У цих трьох класах показані основні методи які відповідають за опрацювання та відображення даних з використання динамічної пам’яті. В усіх таблицях та діаграмах.

Завдяки цих класам та методам відбувається основна робота під «капотом» програми.

Такий підхід до проектування можна назвати терміном «Багатошарова архітектура системи», вона має як свої переваги так і недоліки. Нижче будуть наведені пояснення.

До переваг можливо віднести:

- гарно продумана архітектура допомагає розширювати, змінювати та навіть налагоджувати та тестувати усю систему;
- дає можливість змінювати архітектуру навіть старі версії;
- підвищує швидкість розробки, особливо великого та важкого програмного забезпечення;
- простіша реалізація порівняно з іншими підходами;

До недоліків можливо віднести:

- на реалізацію потрібно набагато більше часу;
- потрібне комплексне тестування усієї системи;
- важко продовжувати розробку програмного забезпечення після певного моменту, відбувається нагромадження усіх компонентів;
- новому розробнику необхідно багато часу, щоб зрозуміти усю систему;

Проектування архітектури системи на пряму впливає на проектування інтерфейсу програмного забезпечення. Проектування інтерфейсу буде виділено в окремий розділ.

Оцінивши усі переваги та недоліки обраного підходу проектування архітектури системи можливо зробити висновки. Що для розробки програмного забезпечення з аналізу використання динамічної пам’яті цей підхід пасує як найліпше. Система не повинна обчислювати дані наприклад на сервері. Не має потреби зберігати великі обсяги інформації та за потреби надавати її користувачу.

3.5 Проектування інтерфейсу користувача

Для проектування інтерфейсу програмного забезпечення для аналізу часових характеристик використання динамічної пам’яті було використано Qt Creator. Оскільки мовою програмування для програми було обрано C++, вибір фрейморку qt – є правильним та доцільним. У Qt Creator є велика

кількість готових шаблонів та макетів для створення необхідного програмного забезпечення. Qt – це фрейморк який використовують для розробки програмного забезпечення використовуючи мову програмування C++.

Для створення інтерфейсу програми було проаналізовано аналоги, визначено їх певні переваги та недоліки. Зрозумівши як інтерфейси аналогічних програм проектували розробники у великих компаніях. Вирішено взяти прототип саме з працюючих програм та доробити його під сформовані вимоги.

Програма включає в себе дві діаграми та чотири таблички. Усі поля табличок та малюнки діаграм заповнюються виключно після збору та аналізу даних. Це необхідно для того, щоб показники з використання динамічної пам’яті були якомога точніше передані.

Усі вікна можливо налаштовувати та редагувати для більш зручного користування. Так для зручності було реалізовано можливість винести вікно з діаграмою або з табличкою наприклад в іншу частину робочої області. Це підвищує комфорт роботи з програмою та позволить не займати багато місця коли не усі дані потрібні до аналізу та використання.

Найкращим способом для проектування інтерфейсу користувача є створення макетів. Завдяки цьому способу можливо помітити помилки у проектуванні інтерфейсу та виправити їх.

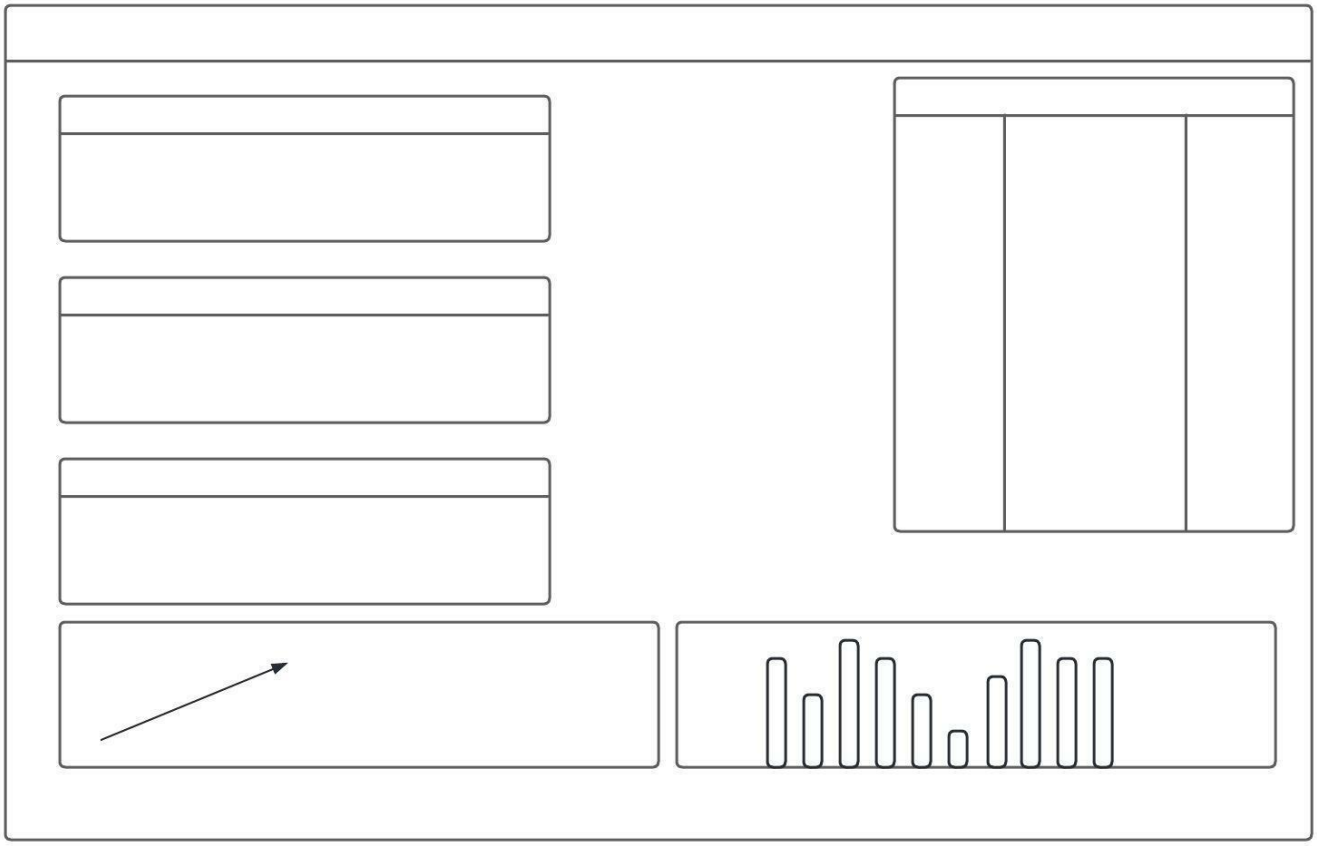

Рисунок 3.8 Макет програми

Qt використовують для створення дуже швидких та високопродуктивних додатків. Разом з мовою програмування C++ можливо отримати високопродуктивну програму з зручним інтерфейсом. Тому вибір технологій при розробці програмного забезпечення впав саме на таке поєднання.

Модулі Qt які були використані під час проектування та створення інтерфейсу користувача:

– QtGUI це компоненти створення інтерфейсів;

– QtWidgets це модуль для роботи із віджетами.

Qt має певні переваги та певні недоліки перед аналогами і нижче пояснено усі тонкощі.

Зручним середовищем для створення інтерфесу є Qt Creator. У цьому середовищі доволі легко розібратись. Для розробки у ньому є усе необхідне, важливі компоненти завжди під рукою. В самому середовищі дуже зручно

проводити тестування програмного забезпечення, тим паче коли мова розробки C++.

Додаткові інструменти допомагають швидко спроектувати та розробити дизайн та зручний інтерфейс програми. Саме тому цей фреймворк був використаний для створення необхідного програмного забезпечення.

Звісно є і певні недоліки які впливали на проектування інтерфесу.

Під час створення інтерфейсу Qt додає багато нових сутностей, звісно що усі вони займають місце. Це у кінцевому результаті впливає на увесь проект, так він може швидко працювати, але він багато важить.

Нижче буде продемонстровано створенні таблиці та діаграми у працюючому програмному забезпечені:

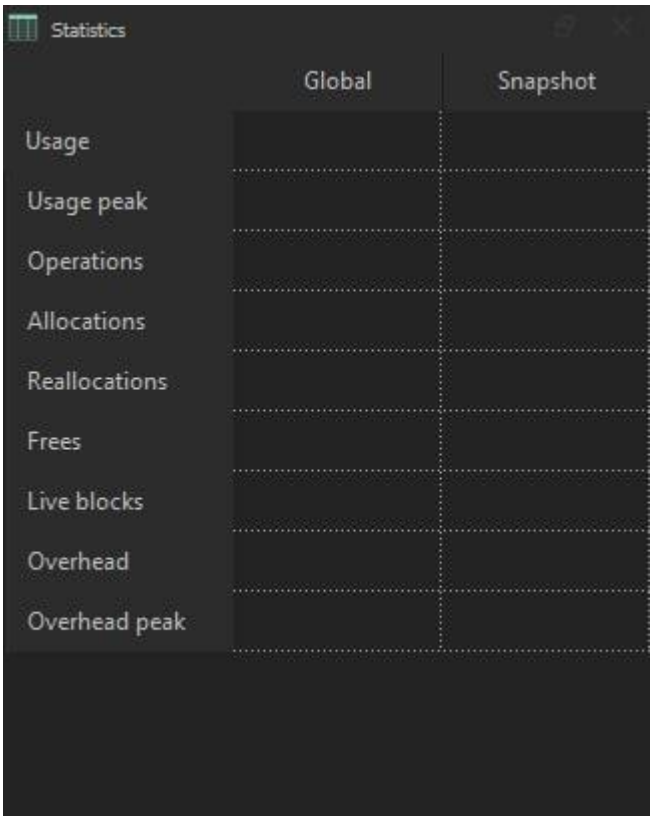

Рисунок 3.9 Табличка з даними про використання динамічної пам’яті

У цій табличці демонструються такі дані з використання динамічної пам’яті:

– застосування;
– пік використання;
– кількість операцій;

– розподілення;
– перерозподіл;
– звільнена пам’ять;
– активні блоки пам’яті;

Це загальна інформація з використання та розподілу динамічної пам’яті.

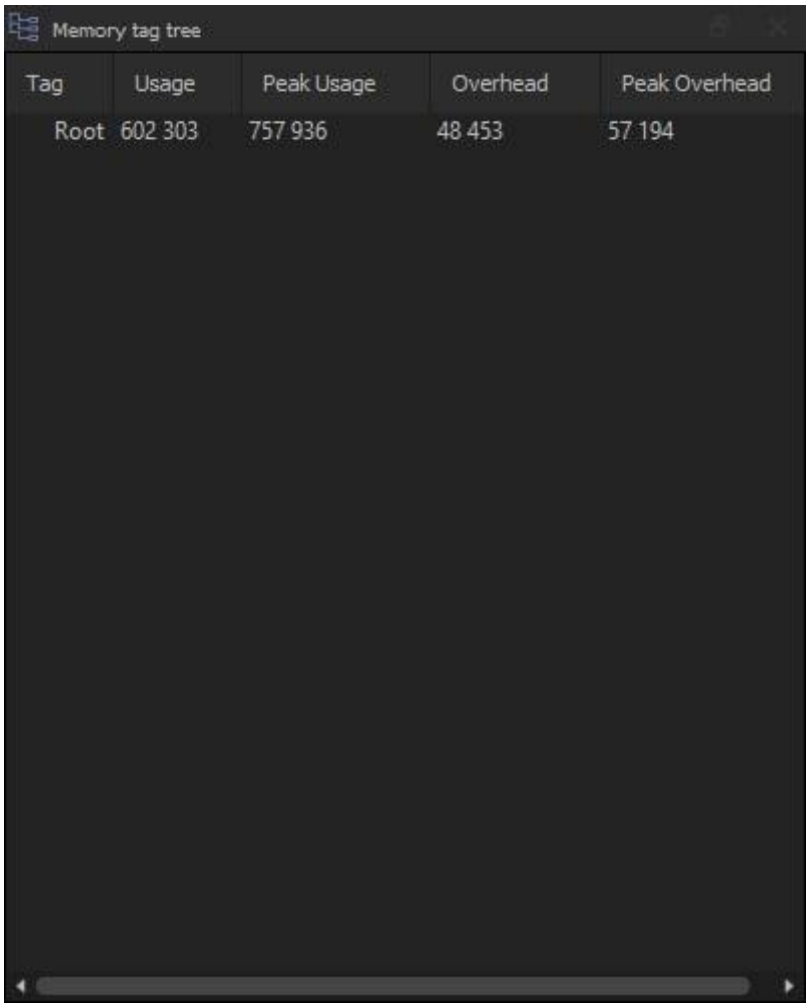

Рисунок 4 Описує дерево тегів;

У цій табличці зібрані дані за допомогою яких можливо проаналізувати дерево тегів. Також тут продемонстровано використання динамічної пам’яті в режимі реального часу.

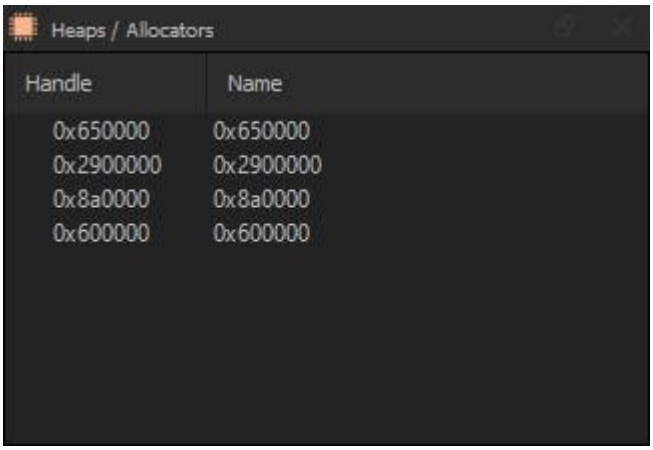

Рисунок 4.1 Heap Allocators

Приклад повного списку з усіх використаних Heap за час роботи та аналізування використання динамічної пам’яті. Heap не має конкретного обмеження на обсяг пам’яті. Пам’ять у Heap виділяється у довільному порядку, за необхідності. Також Heap може бути реалізована з використанням з використанням масиву та дерев. Але навіть маючи такі переваги, швидкість доступу повільніша у порівнянні з Stack.

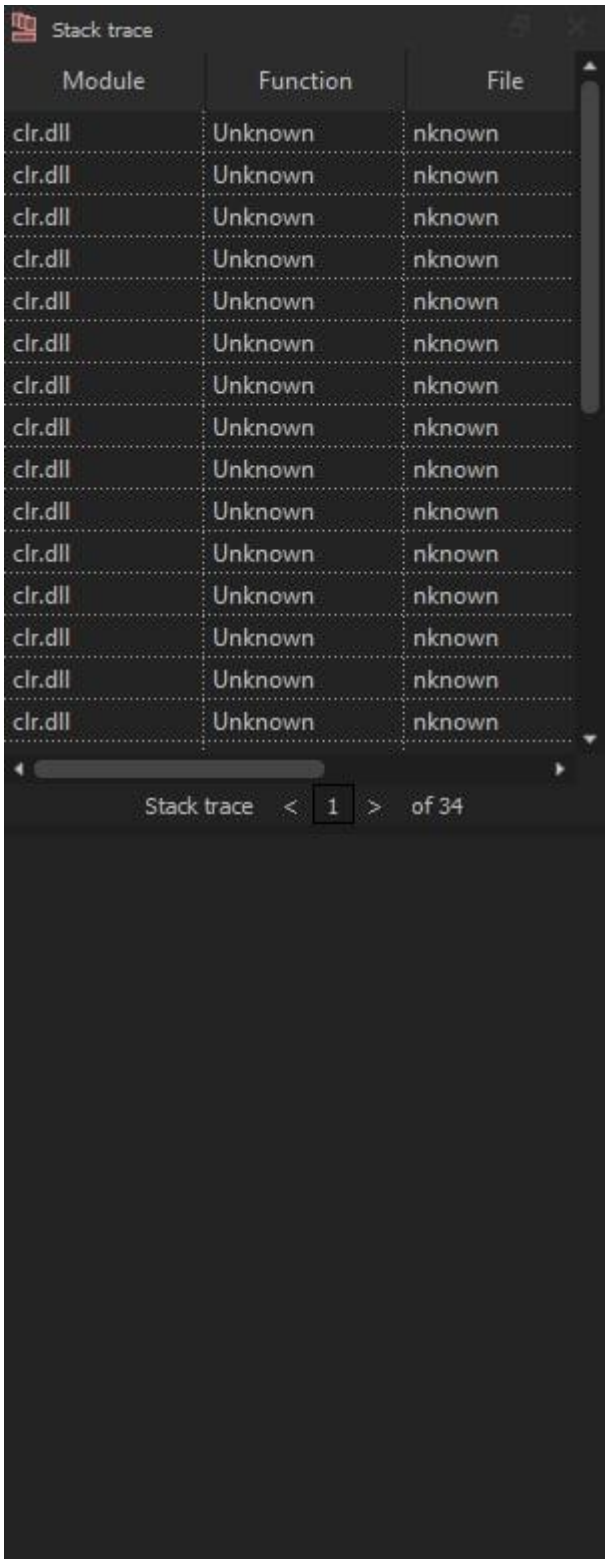

Рисунок 4.3 Stack

Демонструється використання оперативної пам’яті: Stack.

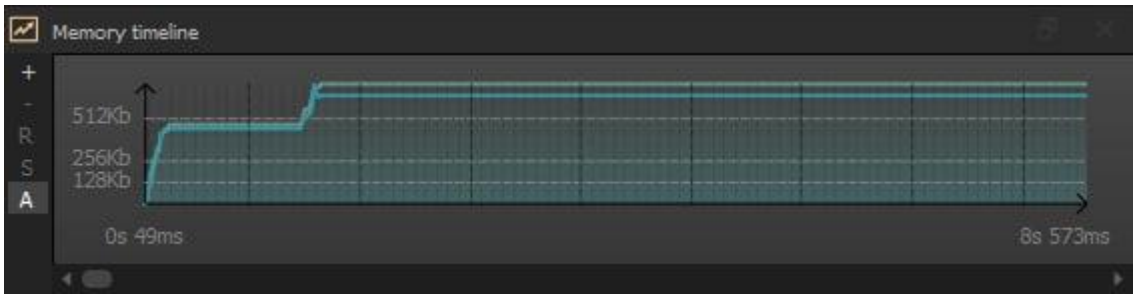

Рисунок 4.4 Memory timeline

Це часова діаграма використання динамічної пам’яті. На цій діаграмі можливо побачити, як виділялась динамічна пам’ять с перебігом часу. Діаграму можливо налаштовувати, збільшувати її, переносити в інший робочій простір(наприклад на інший екран). Можливо обирати певні відрізки часу для повного аналізу використання динамічної пам’яті.

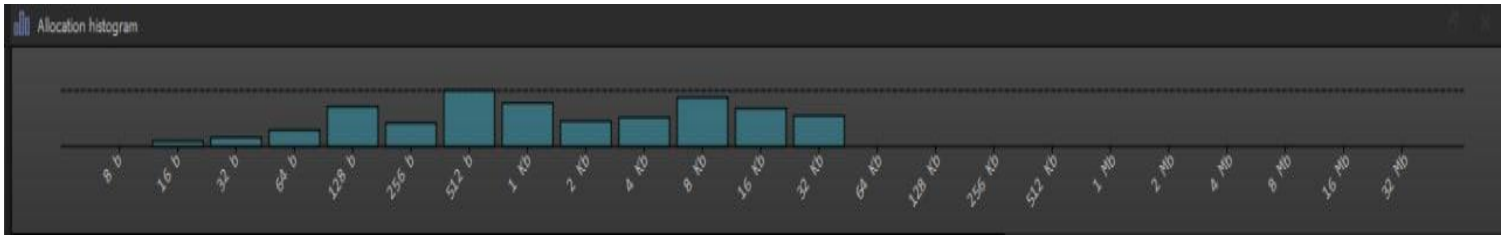

Рисунок 4.5 Allocation

На цій діаграмі можливо побачити розподіл динамічної пам’яті за розміром. Кожне виділення пам’яті відображено в залежності від розміру використаної пам’яті.

Також на цій діаграмі відображується:

– загальна кількість виділень динамічної пам’яті;
– стовпчики які демонструють у якій кількості виділяється пам’ять під певний процес;

Thread ID	Heap	Address	Type	Size
0xd88	0x600000	0x6005b8	Alloc	532
0x3a30	0x600000	0x6027d8	Alloc	532
0x26e0	0x600000	0x6029f8	Alloc	532
0xd88	0x600000	0x602ec0	Free	512
0xd88	0x600000	0x602ec0	Alloc	128
0xd88	0x600000	0x602ec0	Alloc	512
0x21b0	0x600000	0x6025b8	Alloc	532
0xd88	0x600000	0x603188	Alloc	512
0xd88	0x600000	0x603188	Free	512
0xd88	0x600000	0x602e38	Alloc	128
0x1ab4	0x600000	0x600c60	Free	2 048
0x1ab4	0x600000	0x602048	Free	24
0x1ab4	0x600000	0x601fd0	Free	84
0x1ab4	0x600000	0x602030	Free	12
0xd88	0x600000	0x603188	Alloc	512
0xd88	0x600000	0x603188	Free	512
0xd88	0x600000	0x603188	Free	512
0xd88	0x600000	0x603188	Alloc	512
0xd88	0x600000	0x602f48	Alloc	128
0xd88	0x600000	0x602fd0	Alloc	40
0xd88	0x600000	0x603000	Alloc	384

Рисунок 4.6 Демонстрація активних потоків

У цій таблиці користувач може переглянути повний список усіх операцій з пам’яттю. Тут можливо переглянути такі зібрані показники:

– id потоку для якого було виділено пам’ять;
– адресу блоку пам’яті у якому виконувались певні операції в операційній системі Windows;
– розмір блоку пам’яті;
– тип операції з пам’яттю;

Підбиваючи підсумки, можливо сказати, що використовуючи Qt фреймворк було реалізовано доволі зручний інтерфейс. Він не перевантажений, та у той же час дозволяє повність відслідковувати та аналізувати використання динамічної пам’яті в ОС Windows.

Було проведено етап побудови прототипу, це дозволило структурно підійти до створення інтерфейсу користувача.

Інтерфейс був поділений на:

– діаграми;
– таблиці;

Це дозволяє не плутати у якій області програмного забезпечення відображаються необхідні показники. Вікна можливо виносити за межі головного вікна.

У таблицях відображено:

– типи операцій з пам’яттю;
– розміри блоків пам’яті;
– у я кому потоці було виділено динамічну пам’ять;
– перерозподіл динамічної пам’яті.

На діаграмах можливо побачити:

– виділення пам’яті з перебігом часу;
– загальну кількість разів коли було виділено динамічну пам’ять;
– у якій кількості виділяється пам’ять під певний процес;

Підбиваючи загальний підсумок, інтерфейс спроектовано з усіма вимогами до програмного забезпечення.

3.6 Вибір мови програмування

Для розробки програми за допомогою якої можливо аналізувати використання динамічної пам’яті, була обрана мова програмування C++.

Програмісти які використовують C++ залишаються затребуваними завдяки тому що, за допомогою цієї мови програмування можливо створити

майже все. На ньому пишуть бекенд, програми, ігри, драйвери, операційні системи.

C++ 20, остання версія, була визнана Міжнародною організацією зі стандартизації (ISO) повнофункціональною, з можливостями, включаючи модулі, концепції та співпрограми.

Модулі дозволяють програмістам вказувати та використовувати модульні компоненти, тому розробники C++ можуть нарешті відійти від складного, схильного до помилок використання попередньої обробки для складання програми.

Концепції дозволяють програмістам вказувати вимоги шаблону до його аргументів. Це реалізує намір шаблонів підтримувати загальне програмування, роблячи їхні інтерфейси точними. Якість коду буде покращено, а використання шаблонів стане легшим.

C++ – об'єктно-орієнтована мова. Об'єктно-орієнтований підхід забезпечує такі функції, як:

– класи;

– успадкування;

– поліморфізм;

– абстракція даних;

– інкапсуляція;

У C++ більше автономії, коли доходить до управління пам'яттю. Фактично ця мова програмування дозволяє повністю контролювати цей процес. Це можливо завдяки тому, що C++ підтримує динамічне виділення або DMA. Слід зазначити, що така незалежність покладає на користувача велику відповідальність, порівняно з управлінням пам'яттю збирачем сміття.

C++ має стандартну бібліотеку шаблонів Temple (STL). Ця бібліотека підвищує ефективність програмістів, дозволяючи вводити менше рядків коду, швидше писати програми та уникати помилок.

3.7 Аналіз проекту

Продуктивність

Програма була створення на мові програмування C++. Це якісно вплинуло на продуктивність програми під час її роботи. Уся система працює з великою кількістю даних. Їх необхідно зібрати, обробити та вивести в таблицях та діаграмах. Оскільки даних дуже багато, через певний час роботи програми можливе оновлення їх в робочій області. Для того щоб цього не відбувалось краще використовувати не усі таблиці та діаграми. Загалом програма відповідає сучасним вимогам до продуктивності роботи програмного забезпечення.

Зручність використання

Під час проектування та розробки було передбачено багато моментів які якісно вплинули на зручність роботи в програмним забезпеченням. А саме, було обрано гарні та зручні віджети, обрано інформативні діаграми які на сто відсотків демонструють як використовувалась динамічна пам’ять.

Було передбачено можливість налаштовувати усі діаграми та таблички це лише додає зручності програмі. Також було реалізовано можливість виносити необхідні таблиці та діаграми до іншої робочої області (наприклад на інший монітор). Це дозволяє заощадити місце на робочому столі. Оскільки під час проектування та розробки було передбачено та реалізовано багато зручних рішень які позитивно пливають на зручність у використанні програми. Розроблене програмне забезпечення відповідає сучасним вимогам зручності.

Відповідність вимогам

Перед проектуванням та розробкою було сформовано вимоги до програмного забезпечення. Це має бути програма за допомогою якої можливо бачити як використовується динамічна пам’ять. Загалом програма відповідає сформованим вимогам у повному обсязі.

Висновки до розділу 3

У цьому розділі було описано основні етапи розробки програмного забезпечення. А саме, формалізацію задачі, вибір мови програмування, описано усі важливі етапи проектування програми та етапи проектування інтерфейсу користувача. Були створені та описані макети програми, створено блок-схеми для кращого розуміння програмного забезпечення. Також надано детальний опис усіх елементів інтерфейсу, надано пояснення яку інформацію та які дії він демонструє.

Описано функціональну структуру програми та її основні елементи. У результаті можливо зрозуміти усю структуру проекту та структуру інтерфейсу програми.

РОЗДІЛ 4 ТЕСТУВАННЯ ТА НАЛАГОДЖЕННЯ

4.1 Огляд та вибір стратегії тестування

Тестування – це необхідна частина розробки програмного продукту. Основна мета проведення тестів – це забезпечення якості програмного забезпечення. За допомогою тестування можливо перевірити правильність роботи програми, зручність використання, визначити поведінку системи.

Доволі популярними стратегії статичного та структурного тестування. У випадку коли необхідно провести тестування програмного забезпечення яке аналізує використання динамічної пам’яті, ці стратегії повністю підходять

Статичне тестування

За допомого статичного тесту оцінюється якість системи без фактичного запуску цієї системи. Використовуючи статичне тестування було перевірено частини коду та елементи які пов’язані з системою, для того, щоб знайти проблеми якомога раніше. Таке тестування має деякі переваги. Якщо попередня перевірка коду призводить до виявлення помилок, це позбавляє необхідності створювати, встановлювати і запускати систему для пошуку та виправлення помилок. До недоліків можливо віднести необхідність виконати таке тестування в потрібний час.

Під час статичного тестування можливо наприклад перевірити роботу функції яка повертає час роботи потоку який використовує динамічну пам’ять:

```
WINBASEAPI
BOOL
WINAPI
GetThreadTimes(
    _In_ HANDLE hThread,
    _Out_ LPFILETIME lpCreationTime,
    _Out_ LPFILETIME lpExitTime,
    _Out_ LPFILETIME lpKernelTime,
    _Out_ LPFILETIME lpUserTime
    );
```

Рисунок 5.1 Приклад функції;

Структурне тестування

Найкращий спосіб виявити всі помилки – запустити програму на реальних пристроях. Програмне забезпечення має працювати на реальних пристроях, і система має бути запущена повністю, щоб знайти всі помилки. Структурні випробування розробляються з урахуванням структури програмного забезпечення. Їх також можна назвати тестами білої скриньки, тому що вони виконуються тестувальниками, які добре знають програмне забезпечення. Структурні тести найчастіше виконуються на окремих компонентах для виявлення локалізованих помилок.

Оскільки структурне тестування – це тестування програми на реальних пристроях та с повністю запущеною системою. Необхідно сформулювати системні та апаратні вимоги до пристроїв. У наступному розділі вони будуть сформовані.

До недоліків структурного тестування можливо віднести необхідність глибокого розуміння тестованого програмного забезпечення. Загальні переваги тестування програмного продукту є:

– можливість знайти та виправити проблеми у функціональній частині програми до її початку використання кінцевим користувачем;

– можливість оптимізувати код;

– автоматизоване тестування дозволяє зберегти дорогоцінний час;

Таким чином використання будь-якої з стратегій тестування дозволить виявити та виправити проблеми які могли виникнути ще на этапі розробки та проектування програмного продукту

4.2 Опис тестів методами «чорного» та «білого» скриньками

Тестування програмного забезпечення вимагає бути дуже уважним до найдрібніших деталей. Тому тестування проводиться на різних етапах.

Як правило, тестування програмного продукту має такі етапи:

– модульне тестування. На цьому етапі оцінюються окремі компоненти системи, щоб переконатися, що ці компоненти працюють належним чином самі собою;

– інтеграційне тестування. Дозволяє виявити будь-які проблеми у інтерфейсі між модулями та функціями;

– тестування системи. Є завершальним етапом процесу перевірки. На цьому етапі можливо побачити, чи оптимально працює уся система.

Приймальне тестування. Фінальний етап тестування. На цьому етапі стає зрозуміло чи готовий програмний продукт до випуску його для користувачів.

Системні вимоги:

– операційна система Windows 10 або наступні версії;

– наявність встановленого CMake;

Апаратні вимоги:

– достатньо вільної пам’яті комп’ютері для встановлення програми;

– 32 або 64-розрядний процесор з тактовою частотою 1GHz;

– оперативна пам’ять не менше 8Gb.

Тестування проводилось на ноутбуці компанії Lenovo з процесором

Intel® Core i5-10300h, тактовою частотою 4GHz, операційною системою

Windows 10.

Тестування методом «чорної скриньки»

Тестування методом «чорної скриньки» засноване на специфікації або тестуванні поведінки – техніка тестування заснована виключно на роботі з зовнішніми інтерфейсами тестованої системи.

Згідно з ISTQB:

Було проведено тестування, як функціональне, так і нефункціональне, що не передбачає знання внутрішнього пристрою компонента або системи.

Метою є пошук помилок у таких категоріях:

– неправильно реалізовані або відсутні функції;

Загалом усі функції реалізовані правильно.

– помилки інтерфейсу;

Було виявлено не коректне відображення вікна з активними операціями.

– помилки поведінки або недостатня продуктивність системи;

Виявлено невелику затримку у роботі програми при отриманні даних з використання динамічної пам’яті. Причиною є велике навантаження на систему.

Оскільки тестування методом білої скриньки має на увазі ситуацію, що ми не маємо не маємо уявлення про структуру та внутрішній устрій системи. Тому потрібно концентруватись на тому, що робить програма, а не на тому, як вона це робить.

Під час тестування не потрібно звертає уваги на архітектуру системи. Як правило, при виконанні тесту чорної скриньки основна робота є з інтерфейсом системи, надаються вхідні дані і перевіряючи вихідні дані, не знаючи, як і де обробляються вхідні дані.

Провівши тестування методом «чорної скриньки» можливо виділити як переваги так і недоліки такого методу. Але забігаючи наперед можливо зробити висновок, що тестування методом «чорної скриньки» у цьому випадку доцільне.

Переваги тестування методом «чорної скриньки»:

– тестувальникам не потрібно вивчати деталі реалізації системи;

– добре підходить і ефективний для великих сегментів коду;

– чітко відокремлює думку користувача від погляду розробника з допомогою чітко визначених ролей;

– можливо провести таке тестування із середньою кваліфікацією, не знаючи реалізації, мови програмування або операційних систем;

Недоліки у такого методу також є. Але вони не сильно впливають на кінцевий результат розробки програмного забезпечення.

Недоліки тестування методом «чорної скриньки»:

– важко автоматизувати;

– сліпе покриття, оскільки тестер не може орієнтуватися на певні сегменти коду або області, схильні до помилок;

– тестові випадки важко розробити;

– вимагає розстановки пріоритетів, що зазвичай неможливо для перевірки всіх шляхів користувача.

Протилежним до тестування методом «чорної скриньки» є тестування методом «білої скриньки».

Тестування білої скриньки

Тестування методом «білої скриньки» – це детальне дослідження внутрішньої логіки та структури коду. Щоб виконати тестування програми методом білої скриньки, потрібно знати внутрішню роботу коду.

Під час тестування цим методом необхідно зазирнути всередину вихідного коду і з'ясувати, який модуль/фрагмент коду поводиться неналежним чином.

Згідно з ISTQB тестування, засноване на аналізі внутрішньої структури компонента або системи;

Було перевірено так проаналізовано структуру усієї системи, перевірено наявність усіх компонентів.

– тест-дизайн, заснований на техніці білої скриньки – процедура написання або вибору тест-кейсів на основі аналізу внутрішнього пристрою системи або компонента;

Було протестовано основні компоненти програми за допомогою UnitTest:

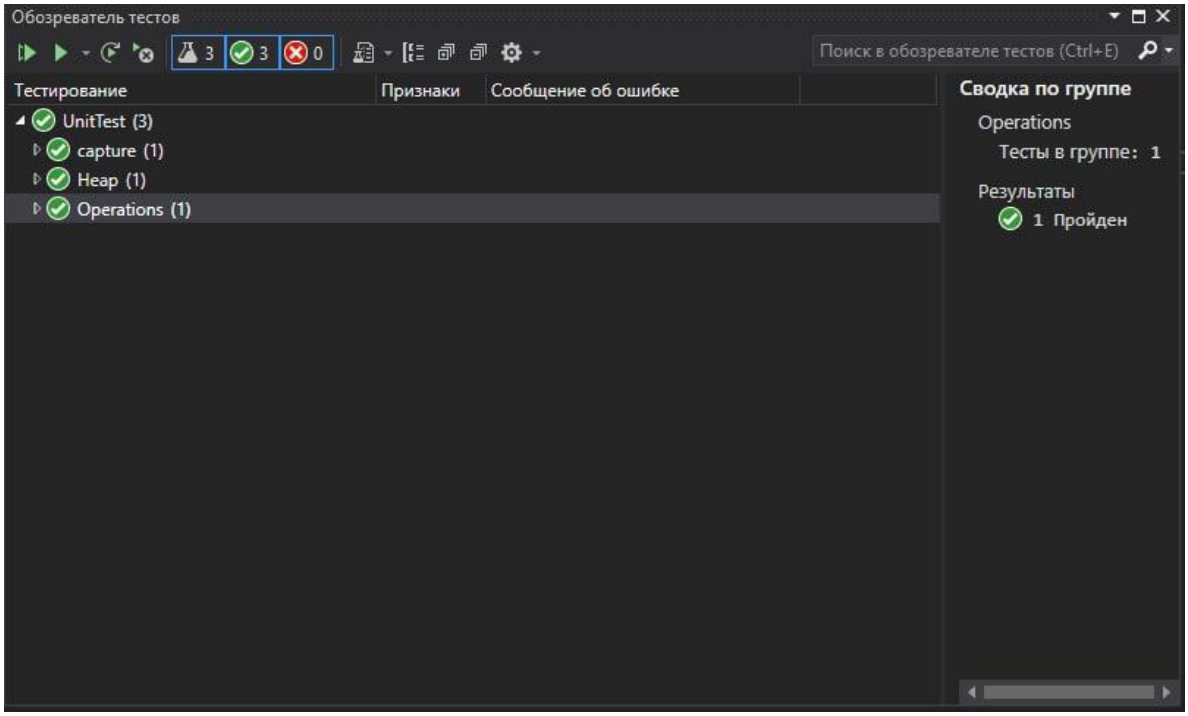

Рис 5.2 Тестування програмного забезпечення;

Тестування методом «білої скриньки» необхідне, тому що, воно допомагає перевірити наступне:

– однією з основних цілей тестування білої скриньки є перевірка роботи програми.

– тестування робиться таким чином, що воно охоплює більшу частину шляхів;

Переваги тестування методом «білої скриньки»:

– допомагає оптимізувати код;

– дозволяє знаходити приховані помилки;

– для тестування цим методом потрібні внутрішні знання, тому це допомагає максимально охопити код.

Недоліки тестування методом «білої скриньки»:

– іноді неможливо заглянути в кожен куточок і знайти приховані помилки, які можуть створити проблеми, оскільки багато шляхів залишаться неперевіреними;

– підтримувати тестування «білої скриньки» складно, тому що для цього потрібні спеціалізовані інструменти, такі як аналізатори коду та засоби налагодження;

– необхідна висока кваліфікація тестувальника.

Проведено тестування цього програмного забезпечення. Було перевірено роботу програмного забезпечення з урахуванням інтерфесу. Перевірено коректне відображення таблиць та діаграм. Результати тестів продемонстровані у таблицях:

Таблиця 4.1 Вибір та налаштування таблиць для аналізу використання динамічної пам’яті

Мета тесту	Перевірити можливість обрати та налаштувати таблиці
Початковий стан	Програма запущена.
Проведення тесту	Обрати та натиснути на іконку яка відповідає за відображення однієї з чотирьох табличок. Налаштувати(якщо можливо) показники які необхідні для аналізу.
Очікуваний результат	Відображення налаштованої таблички.
Наявний результат	Відображено необхідні для аналізу таблички. В яких можливо налаштувати параметри для аналізу використання динамічної пам’яті вОС Windows.

Таблиця 4.2 Вибір та налаштування діаграм для аналізу використання динамічної пам’яті

Мета тесту	Перевірити можливість обрати на налаштувати діграми.
Початковий стан	Програма запущена.
Проведення тесту	Обрати та натиснути на іконку яка відповідає за відображення двох діаграм. Налаштувати (якщо можливо) показники які необхідні для аналізу
Очікуваний результат	В вибраному полі вибереться потрібний елемент і буде готовий до роботи.
Наявний результат	Відображено налаштовані для аналізу використання динамічної пам’яті – діаграми.

Мета тесту	Перевірити можливість перегляду активних операцій які використовують динамічну пам’ять.
Початковий стан	Програма запущена. Запущені активні процеси.
Проведення тесту	Відкрити необхідне вікно у якому буде відображено активні процеси. Переглянути час виконання процесів, та перевірити використання динамічної пам’яті.
Очікуваний результат	Можливість переглядати активні операції та бачити використання динамічної пам’яті.
Наявний результат	В вибраному полі вибереться потрібний елементбуде який буде відображати необхідну інформацію.

Таблиця 4.3 Перевірка можливості переглянути активні операції;

Мета тесту	Перевірити роботу виведення помилки.
Початковий стан	Обрана потрібна таблиця або діаграма.
Вхідні дані	Елементи обраної діаграми або таблиці.
Проведення тесту	Обрати показники які не можливо зібрати.
Наявний результат	У додатковому вікні виведеться повідомлення про помилку.

Таблиця 4.4 Перевірка виводу повідомлення при наявності помилки;

4.3 Аналіз помилок та недоліків програмного забезпечення

Процес тестування програмного забезпечення спрямований на пошук недоліків у існуючому програмному забезпеченні, та на пошук заходів щодо покращення програмного забезпечення з точки зору ефективності, точності та зручності використання.

Типи тестування програмного забезпечення які були використанні:

Рисунок 5.3 Використані типи тестування;

Типи функціонального тестування які використовувались:

– модульне тестування;

Під час розробки програмного забезпечення було проведено модульне тестування. Завдяки чому, вдалось ще на етапі розробки створити максимально оптимізований код. Провівши Юніт-тестування було перевірено працездатність класів та покращити зв’язки певних модулів програми. Проведено тестування таких класів:

– OperationTableSourse;

– BigTableSourse;

– Heap;

Типи нефункціонального тестування які використовувались:

– продуктивність;

Було проведено тестування продуктивності розробленого програмного забезпечення. У випадку розробки системи аналізу використання динамічної пам’яті важливими критеріями продуктивності є: стабільність роботи програми та швидкість відгуку. Оскільки програмне забезпечення було оптимізоване, швидкість відгуку програми досить висока. Вікна, діаграми, таблички швидко реагують на будь які зміни. Стабільність роботи програми при різних навантаженнях зумовлена правильним вибором мови програмування та оптимальним проектуванням системи.

– зручність використання;

Зручність використання програмного забезпечення була досягнута завдяки опитуванню зацікавлених осіб та швидкістю роботи усієї системи. Також, було приділено увагу створенню зручного інтерфейсу та можливості його налаштування.

Аналізуючи розроблене програмне забезпечення можливо виділити певні недоліки які не суттєво впливають на систему:

– можливі деякі похибки при зборі даних з використання динамічної пам’яті;

– велике нагромадження вікон, через що необхідно витрачати час на їх налаштування;

Під час тестування було проаналізовано програмне забезпечення. А саме, було проведено модульні тести, протестовано інтерфейс користувача, функціонал програми та відображення повідомлення у разі виникнення помилки.

Висновки до розділу 4

У цьому розділі було описано певні стратегії тестування програмного забезпечення, а саме стратегії які найдоцільніше застосовувати для програми яку було розроблено.

Також було описано повний цикл тестування програмного забезпечення. А саме наприклад тестування методами «чорного» та «білого» ящика. Для кращого розуміння процес тестування було створено таблички. У яких описано з якого моменту розпочато тестування, описано що користувач зможе побачити виконавши певну дію та описано результат дій користувача.

Було виявлено певні недоліки програми які було описано, та частину з яких було виправлено. Це дозволило оптимізувати код та прискорити час роботи програми. Також завдяки тестування було виявлено та виправлено похибку яка з’являлась під час збору даних. Це могло критично вплинути на продуктивність програми. Ці похибки могли також вплинути на аналіз даних з використання динамічної пам’яті. У результаті тестування програми було виправлено певні помилки у роботі та у відображені певних елементів інтерфейсу.

ВИСНОВКИ

Результатом виконання бакалаврської роботи є десктопний додаток за допомогою якого користувач може бачити використання динамічної пам’яті в ОС Windows. Було проведено повний цикл створення програмного забезпечення для персональних комп’ютерів.

У першому етапі було проведено огляд аналогів, опитування зацікавлених сторін, та постановка задачі. Цей етап був необхідний для формування орієнтирів у подальшій розробці програмного забезпечення.

На наступних етапах було ознайомлення з літературою та поглиблення розуміння стосовно необхідних технологій для розробки програмного забезпечення.

Головними етапами стали «Проектування» та «Тестування та налагодження». У цих етапах дуже докладно описано усі технології які було використано під час розробки. Докладно пояснено як проектувалась система та як проводилось тестування. Для цього було наведена велика кількість малюнків, макетів та таблиць.

Таким чином користувач отримує потужний інструмент для аналізу часових характеристик використання динамічної пам’яті в ОС Windows.

СПИСОК ЛІТЕРАТУРИ

1. ‘C++ Primer’ by Stanley B. Lippman, Josée Lajoie, and Barbara E.[підручник]
2. Тестування [Інтернет ресурс] https://www.browserstack.com/guide/software-testing-strategies-and-approaches.
3. Hands-On High Performance Programming with Qt 5: Build cross-platform ap plications using concurrency, parallel programming, and memory management [підручник]
4. [Інтернет ресурс] https://stepik.org/lesson/341540/step/1
5. [Інтернет ресурс] https://www.educative.io/blog/how-to-design-a-web-application-software-architecture-101#what
6.[Інтернетресурс] https://www.outsystems.com/blog/posts/application-architecture/
7. Stack and Heap [Інтернет ресурс] https://tproger/translations/programming-concepts-stack-and-heap/
8. https://docs.microsoft.com/ru-ru/windows/win32/win7appqual/dynamic-memory
9. [Інтернет ресурс] https://www.qt.io/product/ui-design-tools
10. [Інтернет ресурс] https://www.qt.io/product/features?hsLang=en
11. [Інтернет ресурс] https://coderlessons.com/tutorials/kachestvo-programmnogo-obespecheniia/ruchnoe-testirovanie/chernyi-iashchik-protiv-belaia-korobka
12. [Інтернет ресурс] https://sites.google.com/site/testprogrammprodukt/home/testirovanie-belogo-i-cernogo-asika
13. [Інтернет ресурс] https://otus.ru/nest/post/605/
14. [Інтернет ресурс] https://ravesli.com/urok-105-stek-i-kucha/
15. [Інтернет ресурс] http://web.spt42.ru/index.php/chto-takoe-c-plus-plus
16. [Інтернет ресурс] http://elit-odessa.com.ua/uk/vyrazenie-tipi-danih-v-movi-c-.html

Додатки

ТЕКСТ ПРОГРАМИ

Main.cpp:

```
#ifdef _WIN32
#include <windows.h>
#endif
class DockWidget : public QDockWidget
{
	Q_OBJECT
public:
	DockWidget(const QString& _title,
QWidget* _parent = 0, Qt::WindowFlags
_flags = (Qt::WindowFlags)0)
		: QDockWidget(_title,
_parent, _flags)	{}

public Q_SLOTS:
	void toggleDock(bool)
	{
		setFloating(!isFloating());
	}
};

class ToolButtonHover : public
QToolButton
{
	Q_OBJECT
	QIcon  m_default;
	QIcon  m_hover;

public:
	ToolButtonHover(QIcon _default,
QIcon _hover, QWidget* _parent = 0)
		: QToolButton(_parent)
		, m_default(_default)
		, m_hover(_hover)
	{
	}

	void setDefaultAction(QAction*
_action) { _action->setIcon(m_default);
QToolButton::setDefaultAction(_action); }
	void leaveEvent(QEvent*) {
defaultAction()->setIcon(m_default); }
	void enterEvent(QEvent*) {
defaultAction()->setIcon(m_hover); }
};

class Main : public QMainWindow
{
	Q_OBJECT

private:
	ProjectsManager*
	m_projectsManager;
	QString
	m_watchedFile;
	QTimer*
	m_watchTimer;
	uint64_t
	m_capturePid;
	SymbolStore*
	m_symbolStore;
	GCCSetup*
	m_gccSetup;
	DockWidget*
	m_graphDock;
	DockWidget*
	m_statsDock;
	DockWidget*
	m_histogramDock;
	DockWidget*
	m_tagTreeDock;
	DockWidget*
	m_stackAndSourceDock;
	DockWidget*
	m_heapsDock;
	DockWidget*
	m_modulesDock;
	QProgressBar*
	m_loadingProgressBar;
	QLabel*
	m_statusBarRedDot;
	CentralWidget*
	m_centralWidget;
	QFileDialog*
	m_fileDialog;

	Graph*
	m_graph;
	HistogramWidget*
	m_histogramWidget;
	HeapsWidget*
	m_heapsWidget;
	ModulesWidget*
	m_modulesWidget;
	SourceView*
	m_sourceView;
	StackAndSource*
	m_stackAndSource;
	ExternalEditor*
	m_externalEditor;

	Stats*
	m_stats;
	bool
	m_showWelcomeDialog;
	bool
	m_closeStartPageWidgetOnOpen;

	TagTreeWidget*
	m_tagTree;

public:
	Main(QWidget* _parent = 0,
Qt::WindowFlags _flags =
(Qt::WindowFlags)0);

	void show();
	void setLoadingProgress(float
_progress, const QString &_message);
	void changeEvent(QEvent* _event);
	void closeEvent(QCloseEvent*
_event);
```

```
        void openFileFromPath(const
QString& _file);
        void handleFile(const QString&
_file);

public Q_SLOTS:

        // File
        void openFile();
        void closeFile();
        void openCaptureLocation();
        QString getCaptureLocation();
        void exit();
        // Edit
        void manageProjects();
        void setFilters(bool _filters);
        void setupGCCToolchains();
        // Settings
        void setupSymbols();
        void setupEditor();
        void saveCaptureWindowLayout();
        // Help
        void openDocumentation();
        void about();

        void heapSelected(uint64_t);
        void moduleSelected(void*);
        void graphModified();
        void
setWidgetSources(CaptureContext*
_binView);

        void suicide();
        void showHistogram(bool);
        void showGraph(bool);
        void showStats(bool);
        void showTagTree(bool);
        void showStackTrace(bool);
        void showHeaps(bool);
        void showModules(bool);

        void setStatusBarText(const
QString&);
        void checkCaptureStatus();
        void captureStarted(const
QString&);
        void
captureSetProcessID(uint64_t);

        void setFilteringState(bool,bool);

Q_SIGNALS:
        void binLoaded(bool);
        void setFilterState(bool);
        void setFilterButtonEnabled(bool);

private:
        void showWelcomeDialog();
        void setDockWindowIcon(DockWidget*
_widget, const QString& _icon);
        void setupDockWindows();
        void readSettings();
        void writeSettings();
        static uint32_t
makeVersion(uint8_t _major, uint8_t
_minor, uint8_t _detail);

protected:
        void
dragEnterEvent(QDragEnterEvent* _event);
        void dragMoveEvent(QDragMoveEvent*
_event);
        void
dragLeaveEvent(QDragLeaveEvent* _event);
        void dropEvent(QDropEvent*
_event);
```

BigTable.cpp:

```
class BigTableTableWidget;

class BigTableSource
{
public:
        virtual QStringList
        getHeaderInfo(int32_t& _sortCol,
Qt::SortOrder& _sortOrder, QList<int>&
_widths) = 0;
        virtual uint32_t
        getNumberOfRows() = 0;
        virtual       QString
                getItem(uint32_t _index,
int32_t _column, QColor* _color, bool*
_setColor) = 0;
        virtual void
        getItem(uint32_t _index, void**) =
0;
        virtual Qt::AlignmentFlag
        getAlignment(uint32_t _index) = 0;
        virtual       uint32_t
        getItemIndex(void* _item) = 0;
        virtual void
        sortColumn(uint32_t _colIndex,
Qt::SortOrder _order) = 0;
};

class BigTable : public QWidget
{
        Q_OBJECT

private:
        QStringList
        m_header;
        BigTableTableWidget* m_tree;
        QScrollBar*
        m_scroll;
        BigTableSource*
        m_source;
        int32_t
        m_numColumns;
        int32_t
        m_selectedRows;
        int32_t
        m_firstVisible;
        int32_t
        m_visibleRows;

public:
```

```
	BigTable(QWidget* _parent = 0,
Qt::WindowFlags _flags =
(Qt::WindowFlags)0);

	void changeEvent(QEvent* _event);
	void setSource(BigTableSource*
_src);
	void select(void* _item);
	void updateTable();
	void resetView();
	void ensureSelectionVisible();

	/// QWidget
	void resizeEvent(QResizeEvent*
_event);
	void wheelEvent(QWheelEvent*
_event);
	void keyPressEvent(QKeyEvent*
_event);

	QHeaderView* getHeader() { return
m_tree->horizontalHeader(); }

Q_SIGNALS:
	void itemSelected(void*);
	void itemRightClicked(void*, const
QPoint&);

public Q_SLOTS:
	void scroll(int);
	void
rowSelected(QTableWidgetItem*);
	void
rowRightClick(QTableWidgetItem*, const
QPoint&);
	void sortSelection(int,
Qt::SortOrder);

private:
	Ui::BigTableWidget ui;
};
```

GroupTableSource.cpp:

```
struct GroupColumn
{
	enum Enum
	{
		Type,
		Heap,
		Size,
		Count,
		CountPeak,
		CountPeakPercent,
		Alignment,
		GroupSize,
		GroupPeakSize,
		GroupPeakSizePercent,
		Live,

		ColumnCount =
rtm::MemoryOperationGroup::INDEX_MAPPINGS
	};
};

class GroupTableSource : public
BigTableSource
{
	private:
		CaptureContext*
	m_context;
		GroupList*
	m_list;
		uint32_t
	m_numColumns;
		uint32_t
	m_numRows;


	rtm_vector<rtm::MemoryOperationGro
up*>	m_allGroups;
		const rtm::MemoryStats*
				m_stats;

		GroupMapping
	m_groupMappings[GroupColumn::Colum
nCount];
		GroupMapping*
	m_currentGroupMapping;

		uint32_t
	m_currentColumn;
		Qt::SortOrder m_sortOrder;

	public:

	GroupTableSource(CaptureContext*
_capture, GroupList* _list);
		virtual ~GroupTableSource()
{}

		void prepareData();

		virtual QStringList
	getHeaderInfo(int32_t& _sortCol,
Qt::SortOrder& _sortOrder, QList<int>&
_widths);
		virtual uint32_t
	getNumberOfRows();
		virtual	QString
	getItem(uint32_t _index, int32_t
_column, QColor*, bool*);
		virtual void
	getItem(uint32_t _index, void**);
		virtual Qt::AlignmentFlag
	getAlignment(uint32_t _index);
		virtual uint32_t
	getItemIndex(void* _item);
		virtual void
	sortColumn(uint32_t _columnIndex,
Qt::SortOrder _sortOrder);
		GroupMapping*
	getGroupMapping(int _index) {
return &m_groupMappings[_index]; }

		void saveState(QSettings&
_settings);
};
struct pSortTypeNVC
```

```
{
	rtm_vector<rtm::MemoryOperationGro
up*>* m_allGroups;
	pSortTypeNVC(rtm_vector<rtm::Memor
yOperationGroup*>& _groups) :
m_allGroups(&_groups) {}

	inline bool operator()(const
uint32_t _val1, const uint32_t _val2)
const { return (*m_allGroups)[_val1]-
>m_operations[0]->m_operationType <
(*m_allGroups)[_val2]->m_operations[0]-
>m_operationType; }
};

struct pSortHeapNVC
{
	rtm_vector<rtm::MemoryOperationGro
up*>* m_allGroups;
	pSortHeapNVC(rtm_vector<rtm::Memor
yOperationGroup*>& _groups) :
m_allGroups(&_groups) {}

	inline bool operator()(const
uint32_t _val1, const uint32_t _val2)
const { return (*m_allGroups)[_val1]-
>m_operations[0]->m_allocatorHandle <
(*m_allGroups)[_val2]->m_operations[0]-
>m_allocatorHandle; }
};

struct pSortSizeNVC
{
	rtm_vector<rtm::MemoryOperationGro
up*>* m_allGroups;
	pSortSizeNVC(rtm_vector<rtm::Memor
yOperationGroup*>& _groups) :
m_allGroups(&_groups) {}

	inline bool operator()(const
uint32_t _val1, const uint32_t _val2)
const { return (*m_allGroups)[_val1]-
>m_maxSize< (*m_allGroups)[_val2]-
>m_maxSize; }
};

struct pSortCountNVC
{
	rtm_vector<rtm::MemoryOperationGro
up*>* m_allGroups;
	pSortCountNVC(rtm_vector<rtm::Memo
ryOperationGroup*>& _groups) :
m_allGroups(&_groups) {}

	inline bool operator()(const
uint32_t _val1, const uint32_t _val2)
const { return (*m_allGroups)[_val1]-
>m_count < (*m_allGroups)[_val2]-
>m_count; }
};

struct pSortCountPeakNVC
{
	rtm_vector<rtm::MemoryOperationGro
up*>* m_allGroups;
	pSortCountPeakNVC(rtm_vector<rtm::
MemoryOperationGroup*>& _groups) :
m_allGroups(&_groups) {}

	inline bool operator()(const
uint32_t _val1, const uint32_t _val2)
const { return (*m_allGroups)[_val1]-
>m_liveCountPeak < (*m_allGroups)[_val2]-
>m_liveCountPeak; }
};

struct pSortCountPeakPercentNVC
{
	rtm_vector<rtm::MemoryOperationGro
up*>* m_allGroups;
	pSortCountPeakPercentNVC(rtm_vecto
r<rtm::MemoryOperationGroup*>& _groups) :
m_allGroups(&_groups) {}

	inline bool operator()(const
uint32_t _val1, const uint32_t _val2)
const
	{
		rtm::MemoryOperationGroup*
grp1 = (*m_allGroups)[_val1];
		rtm::MemoryOperationGroup*
grp2 = (*m_allGroups)[_val2];

		return float(grp1-
>m_liveCountPeak * grp2-
>m_liveCountPeakGlobal) < float(grp2-
>m_liveCountPeak * grp1-
>m_liveCountPeakGlobal);
	}
};

struct pSortAlignmentNVC
{
	rtm_vector<rtm::MemoryOperationGro
up*>* m_allGroups;
	pSortAlignmentNVC(rtm_vector<rtm::
MemoryOperationGroup*>& _groups) :
m_allGroups(&_groups) {}

	inline bool operator()(const
uint32_t _val1, const uint32_t _val2)
const { return (*m_allGroups)[_val1]-
>m_operations[0]->m_alignment <
(*m_allGroups)[_val2]->m_operations[0]-
>m_alignment; }
};

struct pSortGroupSizeNVC
{
	rtm_vector<rtm::MemoryOperationGro
up*>* m_allGroups;
	pSortGroupSizeNVC(rtm_vector<rtm::
MemoryOperationGroup*>& _groups) :
m_allGroups(&_groups) {}
```

```
	inline bool operator()(const
uint32_t _val1, const uint32_t _val2)
const
	{
		rtm::MemoryOperationGroup*
grp1 = (*m_allGroups)[_val1];
		rtm::MemoryOperationGroup*
grp2 = (*m_allGroups)[_val2];
		return (grp1->m_liveSize) <
(grp2->m_liveSize);
	}
};

struct pSortGroupSizePeakNVC
{
	rtm_vector<rtm::MemoryOperationGro
up*>* m_allGroups;
	pSortGroupSizePeakNVC(rtm_vector<r
tm::MemoryOperationGroup*>& _groups) :
m_allGroups(&_groups) {}

	inline bool operator()(const
uint32_t _val1, const uint32_t _val2)
const
	{
		rtm::MemoryOperationGroup*
grp1 = (*m_allGroups)[_val1];
		rtm::MemoryOperationGroup*
grp2 = (*m_allGroups)[_val2];
		return (grp1->m_peakSize) <
(grp2->m_peakSize);
	}
};

struct pSortGroupSizePeakPercentNVC
{
	rtm_vector<rtm::MemoryOperationGro
up*>* m_allGroups;
	pSortGroupSizePeakPercentNVC(rtm_v
ector<rtm::MemoryOperationGroup*>&
_groups) : m_allGroups(&_groups) {}

	inline bool operator()(const
uint32_t _val1, const uint32_t _val2)
const
	{
		rtm::MemoryOperationGroup*
grp1 = (*m_allGroups)[_val1];
		rtm::MemoryOperationGroup*
grp2 = (*m_allGroups)[_val2];

		return grp1->m_peakSize *
grp2->m_peakSizeGlobal < grp2->m_peakSize
* grp1->m_peakSizeGlobal;
	}
};

struct pSortLiveNVC
{
	rtm_vector<rtm::MemoryOperationGro
up*>* m_allGroups;
	pSortLiveNVC(rtm_vector<rtm::Memor
yOperationGroup*>& _groups) :
m_allGroups(&_groups) {}

	inline bool operator()(const
uint32_t _val1, const uint32_t _val2)
const { return (*m_allGroups)[_val1]-
>m_liveCount < (*m_allGroups)[_val2]-
>m_liveCount; }
};

struct pSetGroupMappingsNVC
{
	rtm_vector<rtm::MemoryOperationGro
up*>* m_allGroups;
	pSetGroupMappingsNVC(rtm_vector<rt
m::MemoryOperationGroup*>& _groups) :
m_allGroups(&_groups) {}

	inline void operator()(const
GroupMapping* inGroupMapping) const
	{
		const uint32_t col =
inGroupMapping->m_columnIndex;
		const uint32_t num =
(uint32_t)(*m_allGroups).size();
		for (uint32_t i = 0; i<num;
++i)
		{
			uint32_t idx =
inGroupMapping->m_sortedIdx[i];

	rtm::MemoryOperationGroup* group =
(*m_allGroups)[idx];
			group-
>m_indexMappings[col] = i;
		}
	}
};

#endif

GroupTableSource::GroupTableSource(Captur
eContext* _context, GroupList* _list ) :
	m_context(_context),
	m_list(_list)
{
	prepareData();
}

void GroupTableSource::prepareData()
{
	bool filterEnabled = m_list-
>getFilteringState();

	m_stats = &m_context->m_capture-
>getGlobalStats();
	const rtm::MemoryGroupsHashType*
groups = &m_context->m_capture-
>getMemoryGroups();
	if (filterEnabled)
	{
		m_stats = &m_context-
>m_capture->getSnapshotStats();
		groups = &m_context-
>m_capture->getMemoryGroupsFiltered();
	}
```

OperationTableSource.cpp:

```
class Hotspots;
class OperationsList;
class HotspotsWidget;
class StackTreeWidget;
struct CaptureContext;

class BinLoaderView : public QWidget
{
	Q_OBJECT

private:
	QTabWidget*			m_tab;
	TreeMapWidget*
	m_treeMap;
	CaptureContext*
	m_context;
	OperationsList*
	m_operationList;
	GroupList*
	m_groupList;
	HotspotsWidget*
	m_hotspots;
	StackTreeWidget*	m_stackTree;
	OperationsList*
	m_operationListInvalid;
	uint64_t
	m_minTime;
	uint64_t
	m_maxTime;
	uint64_t
	m_currentHeap;
	rdebug::ModuleInfo*
	m_currentModule;
	rtm::StackTrace**
	m_savedStackTraces;
	uint32_t
	m_savedStackTracesCount;
	int
	m_histogramType;
	int
	m_histogramMode;
	bool
	m_histogramPeaks;
	bool
	m_filteringEnabled;

public:
	BinLoaderView(QWidget* _parent =
0, Qt::WindowFlags _flags =
(Qt::WindowFlags)0);
	virtual ~BinLoaderView();

	CaptureContext*
	getContext() { return m_context; }
	void
	setContext(CaptureContext*
_context);

	rtm::StackTrace**
	getSavedStackTraces() { return
m_savedStackTraces; }
	uint32_t
	getSavedStackTracesCount() {
return m_savedStackTracesCount; }

	void		changeEvent(QEvent*
_event);
	void		setHistogramType(int
_type) { m_histogramType = _type; }
	void		setHistogramMode(int
_mode) { m_histogramMode = _mode; }
	void
	setHistogramPeaks(bool _peaks) {
m_histogramPeaks = _peaks; }
	int
	getHistogramType() { return
m_histogramType; }
	int
	getHistogramMode() { return
m_histogramMode; }
	bool		getHistogramPeaks()
{ return m_histogramPeaks; }
	uint64_t	getMinTime() {
return m_minTime; }
	uint64_t	getMaxTime() {
return m_maxTime; }
	void		setMinTime(uint64_t
	_minTime) { m_minTime = _minTime;
}
	void		setMaxTime(uint64_t
	_maxTime) { m_maxTime = _maxTime;
}
	uint64_t	getCurrentHeap() {
return m_currentHeap; }
	void
	setCurrentHeap(uint64_t inHeap) {
m_currentHeap = inHeap; }
	rdebug::ModuleInfo*
getCurrentModule() { return
m_currentModule; }
	void
	setCurrentModule(rdebug::ModuleInf
o* _module) { m_currentModule = _module;
}
	void
	setFilteringEnabled(bool _filter);
	bool
	getFilteringEnabled() const {
return m_filteringEnabled; }

	void readSettings();
	void saveSettings();

public Q_SLOTS:
	void
saveStackTrace(rtm::StackTrace**, int);

Q_SIGNALS:
	void
setStackTrace(rtm::StackTrace**, int);
	void highlightTime(uint64_t);
	void highlightRange(uint64_t,
uint64_t);
	void selectRange(uint64_t,
uint64_t);
```

```
private:
	Ui::BinLoaderView ui;
};
```

Stacktrace.cpp:

```
class QSpinBox;
struct CaptureContext;

class QToolTipper : public QObject
{
	Q_OBJECT

public:
	explicit QToolTipper(QObject*
parent = NULL)
		: QObject(parent) {}

protected:
	bool eventFilter(QObject* obj,
QEvent* event);
};

class StackTrace : public QWidget
{
	Q_OBJECT

private:
	rtm::StackTrace**
	m_currentTrace;
	uint32_t
	m_currentTraceCnt;
	uint32_t
	m_currentTraceIdx;
	QTableWidget*		m_table;
	QMenu*
	m_contextMenu;
	QAction*
	m_actionCopy;
	QAction*
	m_actionCopyAll;
	int
	m_copyIndex;
	QToolButton*		m_buttonDec;
	QToolButton*		m_buttonInc;
	QSpinBox*
	m_spinBox;
	QLabel*
	m_totalTraces;
	CaptureContext*
	m_context;
	rtm::StackTrace*	m_stackTrace;
	QString
	m_selectedFunc;
	QString
	m_settingsGroupName;

public:
	StackTrace(QWidget* _parent = 0,
Qt::WindowFlags _flags =
(Qt::WindowFlags)0);

	void changeEvent(QEvent* _event);
	void
contextMenuEvent(QContextMenuEvent*
_event);
	void setContext(CaptureContext*
_context);
	void clear();
	void updateView();
	void loadState(QSettings&
_settings, const QString& _name, bool
_resetGeometry);
	void saveState(QSettings&
_settings);

public Q_SLOTS:
	void currentCellChanged(int
_currentRow, int _currentColumn, int
_previousRow, int _previousColumn);
	void
setStackTrace(rtm::StackTrace**
_stackTrace, int);
	void incPressed();
	void decPressed();
	void copy();
	void copyAll();
	void copyResetIndex();

Q_SIGNALS:
	void openFile(const QString&
_file, int _row, int _column);

private:
	void setCount(uint32_t _cnt);
	Ui::StackTrace ui;
};
```

HeapsWidget.cpp:

```
struct CaptureContext;

class HeapsWidget : public QWidget
{
	Q_OBJECT

private:
	QTreeWidget*  m_treeWidget;
	QTreeWidgetItem*m_currentItem;
	CaptureContext*	m_context;

public:
	HeapsWidget(QWidget* _parent = 0,
Qt::WindowFlags _flags =
(Qt::WindowFlags)0);

	void changeEvent(QEvent* _event);
	void setContext(CaptureContext*
_capture);
	void setCurrentHeap(uint64_t
_handle);

public Q_SLOTS:
	void itemClicked(QTreeWidgetItem*,
int);

Q_SIGNALS:
	void heapSelected(uint64_t);
```

```
private:
	Ui::HeapsWidget ui;
};
```

BigTable.cpp:

```
class BigTableTableWidget;

class BigTableSource
{
public:
	virtual QStringList
	getHeaderInfo(int32_t& _sortCol,
Qt::SortOrder& _sortOrder, QList<int>&
_widths) = 0;
	virtual uint32_t
	getNumberOfRows() = 0;
	virtual	QString
		getItem(uint32_t _index,
int32_t _column, QColor* _color, bool*
_setColor) = 0;
	virtual void
	getItem(uint32_t _index, void**) =
0;
	virtual Qt::AlignmentFlag
	getAlignment(uint32_t _index) = 0;
	virtual	uint32_t
	getItemIndex(void* _item) = 0;
	virtual void
	sortColumn(uint32_t _colIndex,
Qt::SortOrder _order) = 0;
};

class BigTable : public QWidget
{
	Q_OBJECT

private:
	QStringList
	m_header;
	BigTableTableWidget* m_tree;
	QScrollBar*
	m_scroll;
	BigTableSource*
	m_source;
	int32_t
	m_numColumns;
	int32_t
	m_selectedRows;
	int32_t
	m_firstVisible;
	int32_t
	m_visibleRows;

public:
	BigTable(QWidget* _parent = 0,
Qt::WindowFlags _flags =
(Qt::WindowFlags)0);

	void changeEvent(QEvent* _event);
	void setSource(BigTableSource*
_src);
	void select(void* _item);
	void updateTable();
	void resetView();
	void ensureSelectionVisible();

	/// QWidget
	void resizeEvent(QResizeEvent*
_event);
	void wheelEvent(QWheelEvent*
_event);
	void keyPressEvent(QKeyEvent*
_event);

	QHeaderView* getHeader() { return
m_tree->horizontalHeader(); }

Q_SIGNALS:
	void itemSelected(void*);
	void itemRightClicked(void*, const
QPoint&);

public Q_SLOTS:
	void scroll(int);
	void
rowSelected(QTableWidgetItem*);
	void
rowRightClick(QTableWidgetItem*, const
QPoint&);
	void sortSelection(int,
Qt::SortOrder);

private:
	Ui::BigTableWidget ui;
};
```

Grap.h:

```
lass Graph : public QWidget
{
	Q_OBJECT

private:
	GraphWidget*	m_graph;
	QToolButton*
	m_buttonZoomIn;
	QToolButton*
	m_buttonZoomOut;
	QToolButton*
	m_buttonZoomReset;
	QToolButton*
	m_buttonZoomSelect;
	QToolButton*
	m_buttonAutoZoom;
	QScrollBar*
	m_scroll;
	CaptureContext*m_context;

public:
	Graph(QWidget* _parent = 0,
Qt::WindowFlags _flags = 0);

	void changeEvent(QEvent*
_event);
	void
setContext(CaptureContext* _context,
BinLoaderView* _binView);
	bool isAutoZoomSet() const;
	inline GraphWidget*
getGraphWidget() { return m_graph; }
```

```
public Q_SLOTS:
        void snapshotSelected();
        void zoomChanged();
        void scrollMoved(int);
        void
highlightTime(uint64_t);
        void
highlightRange(uint64_t, uint64_t);

private:
        Ui::Graph ui;
};
```

GramWigd.h:

```
class HistogramWidget : public
QWidget
{
        Q_OBJECT

private:
        QComboBox*
        m_comboType;
        QComboBox*
        m_comboHist;
        QCheckBox*
        m_chkPeaks;
        HistogramView*
        m_histogramView;
        BinLoaderView* m_binView;

public:
        HistogramWidget(QWidget*
_parent = 0, Qt::WindowFlags _flags
= 0);

        void changeEvent(QEvent*
_event);
        void
setContext(CaptureContext* _context,
BinLoaderView* _binView);

public Q_SLOTS:
        void updateUI();
        void displayTypeChanged(int
_index);
        void displayModeChanged(int
_index);
        void showPeaksChanged(int
_state);

Q_SIGNALS:
        void binClicked();

private:
        Ui::HistogramWidgetClass ui;
};
```

Treem.h:

```
class TreeMapWidget : public QWidget
{
        Q_OBJECT

private:
        QGraphicsScene*                 m_sce
        CaptureContext*                 m_con
        TreeMapView*                    m_gra
        TreeMapGraphicsItem*
        m_map;

public:
        TreeMapWidget(QWidget*
_parent = 0, Qt::WindowFlags _flags
= 0);

        void changeEvent(QEvent*
_event);

        void
setContext(CaptureContext*
_context);
        void setFilteringState(bool
_state);

public Q_SLOTS:
        void treeMapTypeChanged(int
_type);

Q_SIGNALS:
        void
setStackTrace(rtm::StackTrace**,
int);

private:
        Ui::TreeMap ui;
};
```

Stactrad.h:

```
class StackTrace : public QWidget
{
        Q_OBJECT

private:
        rtm::StackTrace**
        m_currentTrace;
        uint32_t                        m_cur
        uint32_t                        m_cur
        QTableWidget*
        m_table;
        QToolButton*
        m_buttonDec;
        QToolButton*
        m_buttonInc;
        QSpinBox*                       m_spi
        QLabel*                         m_tot
        CaptureContext*
        m_context;
        rtm::StackTrace*
        m_stackTrace;
        QString                         m_sel

public:
        StackTrace(QWidget* _parent
= 0, Qt::WindowFlags _flags = 0);

        void changeEvent(QEvent*
_event);
```

```
        void
setContext(CaptureContext*
_context);
        void clear();
        void updateView();
        void saveState(QSettings&
_settings);
        void loadState(QSettings&
_settings);

public Q_SLOTS:
        void selected(int _row, int
_column);
        void
setStackTrace(rtm::StackTrace**
_stackTrace, int);
        void incClicked();
        void decClicked();

Q_SIGNALS:
        void openFile(const QString&
_file, int _row, int _column);

private:
        void setCount(uint32_t
_cnt);
        Ui::StackTrace ui;
};
```

Gropswidjet.h:

```
struct GroupMapping
{
        uint32_t
        rtm_vector<rtm::MemoryOperat
ionGroup*>*     m_allGroups;
        rtm_vector<uint32_t>
};

class GroupList : public QWidget
{
        Q_OBJECT

private:
        CaptureContext*
        m_context;
        BigTable*
        GroupTableSource*
        m_tableSource;
        bool
        uint64_t
        QAction*
        QMenu*

        int
        Qt::SortOrder
        QByteArray

public:
        GroupList(QWidget* _parent =
0, Qt::WindowFlags _flags = 0);
        ~GroupList();

        void changeEvent(QEvent*
_event);
```

```
        void
setContext(CaptureContext*
_context);
        void setFilteringState(bool
_state);
        bool getFilteringState()
const;

        void saveState(QSettings&
_settings);
        void loadState(QSettings&
_settings);

Q_SIGNALS:
        void
setStackTrace(rtm::StackTrace**,
int);
        void
usageSortingDone(GroupMapping*);
        void
peakUsageSortingDone(GroupMapping*);
        void
peakCountSortingDone(GroupMapping*);
        void
leaksSortingDone(GroupMapping*);
        void
highlightTime(uint64_t);
        void
highlightRange(uint64_t, uint64_t);
        void selectRange(uint64_t,
uint64_t);

public Q_SLOTS:
        void
selectionChanged(void*);
        void groupRightClick(void*,
const QPoint&);
        void sortingDoneUsage();
        void sortingDonePeakUsage();
        void sortingDonePeakCount();
        void sortingDoneLeaks();
        void selectTriggered();

private:
        Ui::GroupListWidget ui;
};
```

m_groupList;
m_sortedIdx;
m_enableFiltering;
m_lastRange[2];
m_selectAction;
m_contextMenu;
m_savedOrder;

Histogram.cpp:

```
static bool shouldHighlight(QRect&
_rect, QPoint& _highlight, int
_highlightBin)
{
        if (_highlightBin == -1)
                return false;
        return
_rect.contains(_highlight);
}

Histogram::Histogram(HistogramView*
_histogramWidget)
{
        m_histogramWidget      =
_histogramWidget;
```

```
        m_displayMode           =
DisplayMode::Global;
        m_type
        m_showPeaks
        m_highlight
        m_highlightBin          = -1;
}

QRectF Histogram::boundingRect()
const
{
        QSize sz =
m_histogramWidget->size();
        return QRectF(-sz.width()/2,
-sz.height()/2, sz.width(),
sz.height());
}

QPainterPath Histogram::shape()
const
{
    QPainterPath path;
        path.addRect( QRectF(0, 0,
0, 0) );
    return path;
}

QString Histogram::fromVal(int _val)
{
        QString s;
        if (_val < 1024)
                s =
QString::number(_val) + " b";
        else
                if (_val <
1024*1024)
                        s =
QString::number(_val/1024) + " Kb";
                else
                        s =
QString::number(_val/(1024*1024)) +
" Mb";

        while (s.length() < 6)
                s = " " + s;
        return s;
}

uint64_t
Histogram::getMaxValue(rtm::MemorySt
ats& _stats, HistogramType::Enum
_type, bool _usePeak)
{
        uint64_t maxVal = 0;
        int numBins =
rtm::MemoryStats::NUM_HISTOGRAM_BINS
;
        switch (_type)
        {
        case HistogramType::Size:
                for (int i=0;
i<numBins; ++i)
                {
                        maxVal =
    qMax(maxVal,
= HistogramType::Size
= false;
= QPoint(0, 0); // outside
                maxVal = qMax(maxVal,
    _stats.m_histogram[i].m_sizePeak);
                }
                break;

        case
    HistogramType::Overhead:
                for (int i=0;
    i<numBins; ++i)
                {
                        maxVal =
    qMax(maxVal,
    (uint64_t)_stats.m_histogram[i].m_ov
    erhead);
                        if (_usePeak)

                maxVal = qMax(maxVal,
    (uint64_t)_stats.m_histogram[i].m_ov
    erheadPeak);
                }
                break;

        case HistogramType::Count:
                for (int i=0;
    i<numBins; ++i)
                {
                        maxVal =
    qMax(maxVal,
    (uint64_t)_stats.m_histogram[i].m_co
    unt);
                        if (_usePeak)

                maxVal = qMax(maxVal,
    (uint64_t)_stats.m_histogram[i].m_co
    untPeak);
                }
                break;
        }
        return maxVal;
}

uint64_t
Histogram::getValue(rtm::MemoryStats
& _stats, HistogramType::Enum _type,
int _bin, bool _peak)
{
        switch (_type)
        {
        case HistogramType::Size:
                if (_peak)
                        return
_stats.m_histogram[_bin].m_sizePeak;
                else
                        return
_stats.m_histogram[_bin].m_size;
                break;
```

```
        case
HistogramType::Overhead:
                if (_peak)
                        return
(uint64_t)_stats.m_histogram[_bin].m
_overheadPeak;
                else
                        return
(uint64_t)_stats.m_histogram[_bin].m
_overhead;
                break;

        case HistogramType::Count:
                if (_peak)
                        return
(uint64_t)_stats.m_histogram[_bin].m
_countPeak;
                else
                        return
(uint64_t)_stats.m_histogram[_bin].m
_count;
                break;
        }
        return 0;
}

QString
Histogram::getTypeString(HistogramTy
pe::Enum _type, uint64_t _val, bool
_peak)
{
        QString ret;
        switch (_type)
        {
        case HistogramType::Size:
                if (_val == 1)
                        ret = "1 " +
QObject::tr("byte used");
                else
                        ret =
m_locale.toString(qulonglong(_val))
+ " " + QObject::tr("bytes used");
                break;

        case
HistogramType::Overhead:
                ret =
m_locale.toString(qulonglong(_val))
+ " " + QObject::tr("bytes of
overhead");
                break;

        case HistogramType::Count:
                if (_val == 1)
                        ret = "1 " +
QObject::tr("allocation");
                else
                        ret =
m_locale.toString(qulonglong(_val))
+ " " + QObject::tr("allocations");
                break;
        }
        if (_peak)
                ret += " " +
QObject::tr("at peak");

        return ret;
}

static QColor boostColor(const
QColor& _color, bool _boost)
{
        static int boost = 60;
        if (_boost)
                return QColor(
        min(_color.red() + boost,
255),



        return _color;
}

void Histogram::paint(QPainter*
_painter, const
QStyleOptionGraphicsItem* _option,
QWidget* _widget)
{
        RTM_UNUSED(_option);
        RTM_UNUSED(_widget);
        CaptureContext* ctx =
m_histogramWidget->getContext();
        if (!ctx)
                return;

        QRect rect =
m_histogramWidget->getDrawRect();
        int left = rect.x();
        int top = rect.y();
        int right = rect.width() +
rect.x();
        int bottom = rect.height() +
rect.y();

        _painter-
>setPen(QPen(Qt::black, 1.0,
Qt::DashLine, Qt::RoundCap,
Qt::RoundJoin));
        _painter-
>drawLine(left,top,right,top);
        _painter-
>setPen(QPen(Qt::black, 1.0,
Qt::SolidLine, Qt::RoundCap,
Qt::RoundJoin));
        _painter-
>drawLine(left,bottom,right,bottom);

        QFont
font("Consolas",8,QFont::Normal);
        _painter->setFont(font);

        int numBins =
rtm::MemoryStats::NUM_HISTOGRAM_BINS
;
```

```
        int delta = (right-left) /
(numBins+1);

        QRectF boundRect =
boundingRect();
        int deltaW =
(boundRect.width() - (right-
left))/2;
        int deltaH =
(boundRect.height() - (bottom-
top))/2;

        int currSize =
rtm::MemoryStats::MIN_HISTOGRAM_SIZE
;
        int currPos = deltaW +
delta;

        uint32_t selectedBin = ctx-
>m_capture->getSelectHistogramBin();

        for (int i=0; i<numBins;
++i)
        {
                _painter->save();
                if (i ==
static_cast<int>(selectedBin))
                        _painter-
>setPen(QPen(QColor(50+60, 150+60,
170+60), 1.0, Qt::SolidLine,
Qt::RoundCap, Qt::RoundJoin));
                else
                        _painter-
>setPen(QPen(Qt::lightGray, 1.0,
Qt::SolidLine, Qt::RoundCap,
Qt::RoundJoin));

                QTransform tr;
                tr.translate(currPos
- 15, boundRect.height() - deltaH +
21);
                tr.rotate(-45.0f);
                _painter-
>setTransform(tr);
                _painter-
>drawText(0,0,fromVal(currSize));
                _painter->restore();
                _painter-
>drawLine(QPoint(left+currPos-
deltaW, bottom),
QPoint(left+currPos-deltaW,
bottom+3));
                currPos += delta;
                currSize <<= 1;
      }
int barsPerBin = 1;
        if (m_showPeaks)
                barsPerBin = 2;
        if (m_displayMode ==
DisplayMode::Both)
                barsPerBin *= 2;

        int thickness = (delta & ~1)
- 6;

        while (thickness %
barsPerBin)
                thickness -= 2;

        rtm::MemoryStats statsGlobal   = ctx
        rtm::MemoryStats
statsSnapshot   = ctx->m_capture-
>getSnapshotStats();

        QColor globalCol(50, 150,
170, 138);
        QColor globalColPeak(50+60,
150+60, 170+60, 111);
        QColor snapshotCol(90, 120,
90, 138);
        QColor
snapshotColPeak(90+60, 120+60,
90+60, 111);

        m_toolTips.clear();

        switch (m_displayMode)
        {
        case DisplayMode::Global:
        case DisplayMode::Snapshot:
                {
                        QColor* col
= &globalCol;
                        QColor*
colPeak   = &globalColPeak;


        rtm::MemoryStats* stats =
&statsGlobal;
                        QString
statsType = "Global:\n";
                        if
(m_displayMode ==
DisplayMode::Snapshot)
                        {

        statsType = "Snapshot:\n";
                                stats
= &statsSnapshot;
                                col
= &snapshotCol;

        colPeak   = &snapshotColPeak;
                        }

                        uint64_t
maxVal =
getMaxValue(*stats,m_type,false);
                        uint64_t
maxValPeak =
getMaxValue(*stats,m_type,true);

                        int position
= left + delta;
```

```
                        for (int i=0;
i<numBins; ++i)
                        {
                                if
(m_showPeaks)
                                {
                                        uint64_t binVal = getValue(*stats, m_type,i,false);
                                        uint64_t binHeight = maxVal ? ((bottom - top) * binVal) / maxVal : 0;
                                        if (binHeight != 0)
                                        {
                                                QRect rect1(position-thickness/2,bottom-binHeight,thickness,binHeight);
                                                _painter->setBrush(boostColor(*col, uint64_t(255*0.5)));
                                                _painter->drawRect(rect1);
                                                m_toolTips.push_back(HistogramToolTip(rect1,binVal,binHeight));
                                        }

                                        binVal = getValue(*stats, m_type,i,true);
                                        binHeight = maxValPeak ? ((bottom - top) * binVal) / maxValPeak : 0;
                                        if (binHeight != 0)
                                        {
                                                QRect rect2(position,bottom-binHeight,thickness,binHeight);
                                                _painter->setBrush(boostColor(*col, uint64_t(255*0.25)));
                                                _painter->drawRect(QRect(position,bottom-binHeight,thickness,binHeight));
                                                m_toolTips.push_back(HistogramToolTip(rect2,binVal,binHeight));
                                        }

                                        position += delta;
                                }
                                else
                                {
                                        uint64_t binVal = getValue(*stats, m_type,i,false);
                                        uint64_t binHeight = maxVal ? ((bottom - top) * binVal) / maxVal : 0;
                                        if (binHeight != 0)
                                        {
                                                QRect rect1(position-thickness/2,bottom-binHeight,thickness,binHeight);
                                                _painter->setBrush(boostColor(*col, uint64_t(255*0.5)));
                                                _painter->drawRect(rect1);
                                                m_toolTips.push_back(HistogramToolTip(rect1,binVal,binHeight));
                                        }

                                        position += delta;
                                }
                        }
                }
                break;

        case DisplayMode::Both:
                {
                        uint64_t
maxValGlobal            =
getMaxValue(statsGlobal,m_type,false
);
                        uint64_t
maxValPeakGlobal        =
getMaxValue(statsGlobal,m_type,true)
;
                        uint64_t
maxValSnapshot          =
getMaxValue(statsSnapshot,m_type,fal
se);
                        uint64_t
maxValPeakSnapshot      =
        getMaxValue(statsSnapshot,m_type,tru
        e);
                                int position
        = left + delta;
                                for (int i=0;
        i<numBins; ++i)
        {
                uint64_t binVal = getValue(*stats, m_type,i,false);
                uint64_t binHeight = maxVal ? ((bottom - top) *
                if (binHeight != 0)
                {
                        QRect rect1(position-thickness/2,bottom-
                        _painter->setBrush(boostColor(*col, uint6
                        _painter->drawRect(rect1);
                        m_toolTips.push_back(HistogramToolTip(re
                }

                binVal = getValue(*stats, m_type,i,true);
                binHeight = maxValPeak ? ((bottom - top) * binV
                if (binHeight != 0)
                {
                        QRect rect2(position,bottom-binHeight,th
                        _painter->setBrush(boostColor(*col, uint6
                        _painter->drawRect(QRect(position,bottom-
                        m_toolTips.push_back(HistogramToolTip(re
                }

                position += delta;
        }
        else
        {
                uint64_t binVal = getValue(*stats, m_type,i,false);
                uint64_t binHeight = maxVal ? ((bottom - top) * binVal) / maxVal : 0;
                if (binHeight != 0)
                {
                        QRect rect1(position-thickness/2,bottom-binHeight,thickness,binHeight);
                        _painter->setBrush(boostColor(*col, uint64_t(255*0.5)));
                        _painter->drawRect(rect1);
                        m_toolTips.push_back(HistogramToolTip(rect1,binVal,binHeight));
                }

                position += delta;
        }
```

```
                                }
                        }
                }
                break;
        }
}
```

Treemap.ccp:

```
TreeMapWidget::TreeMapWidget(QWidget
* _parent, Qt::WindowFlags _flags) :
        QWidget(_parent, _flags)
{
        ui.setupUi(this);
        m_graphicsView          =
findChild<TreeMapView*>("graphicsVie
w");
        m_context               = NULL;
        m_map                   = NULL;

        QComboBox* cb =
findChild<QComboBox*>("comboBoxType"
);
        connect(cb,
SIGNAL(currentIndexChanged(int)),
this,
SLOT(treeMapTypeChanged(int)));

        m_scene = new
QGraphicsScene(this);
    m_scene-
>setItemIndexMethod(QGraphicsScene::
NoIndex);

    m_graphicsView-
>setScene(m_scene);
    m_graphicsView-
>setCacheMode(QGraphicsView::CacheBa
ckground);
    m_graphicsView-
>setViewportUpdateMode(QGraphicsView
::FullViewportUpdate);
    m_graphicsView-
>setRenderHint(QPainter::Antialiasin
g);
    m_graphicsView-
>scale(qreal(1.0), qreal(1.0));
        connect(m_graphicsView,
SIGNAL(setStackTrace(rtm::StackTrace
**,int)), this,
SIGNAL(setStackTrace(rtm::StackTrace
**,int)));
}
```

```
}
binVal = getValue(statsSnapshot, m_type,i,false
binHeight = maxValSnapshot ? (bottom - top) *
if (binHeight != 0)
{
        QRect rect2(posit, bottom - binHeight,t
;       _painter->setBrush(boostColor(snapshotC
        _painter->drawRect(rect2);
}
position += delta;
```

```
void TreeMapWidget::changeEvent(QEvent* _event)
{
        QWidget::changeEvent(_event);
        if (_event->type() == QEvent::LanguageChange)
                ui.retranslateUi(this);
}
```

```
void
TreeMapWidget::setContext(CaptureCon
text* _context)
{
        m_context = _context;
        m_graphicsView-
>setContext(_context);
        m_map = new
TreeMapGraphicsItem(m_graphicsView,
_context);
        m_scene->addItem(m_map);
}

void
TreeMapWidget::setFilteringState(boo
l _state)
{
        RTM_UNUSED(_state);
        m_graphicsView-
>setMapType(m_graphicsView-
>getMapType());
}

void
TreeMapWidget::treeMapTypeChanged(in
t _type)
{
        m_graphicsView-
>setMapType(_type);
}
```

HeapWidget.cpp :

```
HeapsWidget::HeapsWidget(QWidget*
_parent, Qt::WindowFlags _flags) :
        QWidget(_parent, _flags)
{
        ui.setupUi(this);

        m_treeWidget =
findChild<QTreeWidget*>("treeWidget"
);
        connect(m_treeWidget,
SIGNAL(currentItemChanged(QTreeWidge
tItem*,QTreeWidgetItem*)), this,
SLOT(ItemChanged(QTreeWidgetItem*,QT
reeWidgetItem*)));
}
```

```
void
HeapsWidget::changeEvent(QEvent*
_event)
{
        QWidget::changeEvent(_event)
;
        if (_event->type() ==
QEvent::LanguageChange)

        ui.retranslateUi(this);
}

void
HeapsWidget::setContext(CaptureConte
xt* _context)
{
        if (m_context == _context)
                return;

        m_context = _context;

        if (m_context)
        {
                m_treeWidget-
>clear();
                rtm::HeapsType&
heaps = m_context->m_capture-
>getHeaps();

        rtm::HeapsType::iterator it
= heaps.begin();

        rtm::HeapsType::iterator end
= heaps.end();

                QTreeWidgetItem*
item = new
QTreeWidgetItem(QStringList()


                item-
>setFirstColumnSpanned(true);
                item->setData(0,
Qt::UserRole, (qlonglong)-1);
                m_treeWidget-
>addTopLevelItem(item);


                while (it != end)
                {
                        item = new
QTreeWidgetItem(QStringList()


                        item-
>setData(0, Qt::UserRole,
(qlonglong)it->first);
                        m_treeWidget-
>addTopLevelItem(item);
                        ++it;
                }
        }
        else
                m_treeWidget-
>clear();
}

void
HeapsWidget::ItemChanged(QTreeWidget
Item* _currentItem, QTreeWidgetItem*
_item)
{
        RTM_UNUSED(_item);
        if (!_currentItem)
                return;
        uint64_t handle =
(uint64_t)_currentItem-
>data(0,Qt::UserRole).toLongLong();
        emit heapSelected(handle);
}

void
HeapsWidget::setCurrentHeap(uint64_t
_handle)
{
        QTreeWidgetItemIterator
it(m_treeWidget);
        while (*it)
        {
                if ((*it)-
>data(0,Qt::UserRole).toULongLong()
== _handle)
                {
                        (*it)-
>setSelected(true);
                        return;
                }
                ++it;
        }
}
```

Graps.cpp :

```
                        << ""
                        << QString("All a

#define REALTIME_UPDATE 1

GraphWidget::GraphWidget(QWidget*
_parent) :
        QGraphicsView(_parent)
{
        m_scene = new
QGraphicsScene(this);
    m_scene-
>setItemIndexMethod(QGraphicsScene::
NoIndex);
                                << ("0x"
    setScene(m_scene);          << QStrin
    setCacheMode(CacheBackground);

setViewportUpdateMode(FullViewportUp
date);

setRenderHint(QPainter::Antialiasing
);
        setMouseTracking(true);
```

```
        m_minTime                   {    = 0;
        m_maxTime                       = 0;m_graph = _graph;
        m_hightlightTime        = (uint64_t)-1;Q_FOREACH( GraphCurve* it,
        m_hightlightTimeEnd     = (uint64_t)-1;m_curves){
        m_hightlightIntensity  =                    it-
0.f;                                    >setGraph(m_graph);
        m_highlightAnimation   =                }
NULL;                                   }
        m_LButtonDown           = false;
        m_isDragging            = false;
        m_RButtonDown           = false;GraphWidget::setMinTime(uint64_t
        m_isPanning                 _minTime)false;
        m_hoverMarkerTime       = 0;{
        m_markerSelectFromTime =            m_minTime = _minTime;
(uint64_t)-1;                       invalidateScene();
        m_markerSelectToTime   =            emit minMaxChanged();
(uint64_t)-1;                       }
        m_context                       = NULL;
        m_inContextMenu         = false;
                                    GraphWidget::setMaxTime(uint64_t
        GraphCurve* curve = new     _maxTime)
GraphCurve(this);                   {
        curve->parentResized();             m_maxTime = _maxTime;
        m_curves.append(curve);             invalidateScene();
        m_scene->addItem(curve);            emit minMaxChanged();
                                    }
        m_markers = new
GraphMarkers(this);                 void
        m_markers->parentResized();GraphWidget::setMinMaxTime(uint64_t
        m_scene->addItem(m_markers);_minTime, uint64_t _maxTime)
                                    {
        m_grid = new                        m_minTime = _minTime;
GraphGrid(this,curve);              m_maxTime = _maxTime;
        m_grid->parentResized();            invalidateScene();
        m_scene->addItem(m_grid);   }

        m_select = new              void
GraphSelect(this);                  GraphWidget::setContext(CaptureConte
        m_select->parentResized();xt* _context, BinLoaderView* _view)
        m_scene->addItem(m_select);{
                                            m_context = _context;
        fitInView(m_scene-                  if (_context == NULL)
>sceneRect());                                  setToolTip(tr(""));
                                            else
        createCustomContextMenu();          {
}
                                                m_minTime = _view-
void                                >getMinTime();
GraphWidget::changeEvent(QEvent*            m_maxTime = _view-
_event)                             >getMaxTime();
{
        QWidget::changeEvent(_event)        if ((m_context-
;                                   >m_capture->getMinTime() ==
        if (_event->type() ==       m_context->m_capture-
QEvent::LanguageChange)             >getSnapshotTimeMin()) &&
        {                                       (m_context-
                                    >m_capture->getMaxTime() ==
        createCustomContextMenu();  m_context->m_capture-
        }                           >getSnapshotTimeMax()))
}                                               m_select-
                                    >setSelectRange(0, 0);
void GraphWidget::setGraph(Graph*           else
_graph)
```

```
                        m_select-
>setSelectRange(m_context-
>m_capture->getSnapshotTimeMin(),
m_context->m_capture-
>getSnapshotTimeMax());
        }
        invalidateScene();
}

void
GraphWidget::drawBackground(QPainter
* _painter, const QRectF& _rect)
{
        RTM_UNUSED(_rect);
        QRectF fullRect =
mapToScene(viewport()-
>geometry()).boundingRect();

        QLinearGradient
gradient(fullRect.topLeft(),
fullRect.bottomLeft());
    gradient.setColorAt(0,
QColor(Qt::darkGray).darker(180));
    gradient.setColorAt(1,
QColor(Qt::darkGray).darker(230));
    _painter->setBrush(gradient);
    _painter->drawRect(fullRect);
}

QRect GraphWidget::getDrawRect()
const
{
        QSize sz = size();
        int hWidth      =
sz.width()/2;
        int hHeight     =
sz.height()/2;

        int left        = -hWidth  +
GraphWidget::s_marginLeft;
        int right       =  hWidth  -
GraphWidget::s_marginRight;
        int top         = -hHeight +
GraphWidget::s_marginTop;
        int bottom      =  hHeight -
GraphWidget::s_marginBottom;

        return QRect(left,top,right-
left,bottom-top);
}

uint64_t
GraphWidget::mapPosToTime(int _x)
const
{
        if (!m_context)
                return 0;

        QPointF scenePos =
mapToScene(QPoint(_x,0));
        QRectF drawRect =
getDrawRect();
        scenePos.setX(qMax(scenePos.
x(), drawRect.x()));
        scenePos.setX(qMin(scenePos.
x(), drawRect.x() +
drawRect.width()));

        float offset = scenePos.x()
- float(drawRect.x());
        float w = drawRect.width();
        uint64_t time = ((offset) *
(m_maxTime - m_minTime)) / w;
        time += m_minTime;
        return time;
}

int
        GraphWidget::mapTimeToPos(ui
nt64_t _x) const
{
        if (!m_context)
                return 0;

        QRectF drawRect =
getDrawRect();
        uint64_t offset = ((_x -
m_minTime) * drawRect.width()) /
(m_maxTime - m_minTime);

        return drawRect.x() +
offset;
}

void
GraphWidget::highlightTime(uint64_t
_time)
{
        if (!m_context)
                return;

        if (_time < m_context-
>m_capture->getMinTime())
                return;
        if (_time > m_context-
>m_capture->getMaxTime())
                return;

        m_hightlightTime                = _ti
        m_hightlightTimeEnd             = _ti

        animateHighlight();
}

void
GraphWidget::highlightRange(uint64_t
_minTime, uint64_t _maxTime)
{
        if (!m_context)
                return;

        if (_minTime > _maxTime)
                std::swap(_minTime,
_maxTime);
```

```
        if (_maxTime < m_context->m_capture->getMinTime())
                return;
        if (_minTime > m_context->m_capture->getMaxTime())
                return;

        if (_minTime < m_context->m_capture->getMinTime())
                _minTime = m_context->m_capture->getMinTime();
        if (_maxTime > m_context->m_capture->getMaxTime())
                _maxTime = m_context->m_capture->getMaxTime();

        m_hightlightTime = _minTime;
        m_hightlightTimeEnd = _maxTime;

        animateHighlight();
}

void GraphWidget::animateHighlight()
{
        if (m_highlightAnimation != NULL)

        m_highlightAnimation->stop();

        m_hightlightIntensity  = 1.0f;

        m_highlightAnimation = new QPropertyAnimation(this, "highlightIntensity");
        connect(m_highlightAnimation, SIGNAL(valueChanged(const QVariant&)), this, SLOT(zoomAnimEvent()));

        m_highlightAnimation->setDuration(666);
        m_highlightAnimation->setStartValue(1.0f);
        m_highlightAnimation->setEndValue(0.0f);
        m_highlightAnimation->start();
}

void GraphWidget::selectFromTimes(uint64_t _t1, uint64_t _t2)
{
        uint64_t mn = qMin(_t1,_t2);
        uint64_t mx = qMax(_t1,_t2);
        m_context->m_capture->setSnapshot(mn, mx);
        m_select->setSelectRange(mn, mx);
        emit snapshotSelected();
        invalidateScene();
        m_markerSelectFromTime = (uint64_t)-1;
        m_markerSelectToTime   = (uint64_t)-1;
        m_actionZoomToSelection->setEnabled(true);
        m_actionSnapSelectionToMarker->setEnabled(true);
}

void GraphWidget::animateRange(uint64_t _min, uint64_t _max)
{
        QPropertyAnimation* animation = new QPropertyAnimation(this, "minTime");
        animation->setDuration(150);
        animation->setStartValue(qulonglong(m_minTime));
        animation->setEndValue(qulonglong(_min));
        animation->start();

        connect(animation, SIGNAL(valueChanged(const QVariant&)), this, SLOT(zoomAnimEvent()));

        animation = new QPropertyAnimation(this, "maxTime");
        animation->setDuration(150);
        animation->setStartValue(qulonglong(m_maxTime));
        animation->setEndValue(qulonglong(_max));
        animation->start();
}

void GraphWidget::zoomIn()
{
        zoomIn(UINT64_MAX);
}

void GraphWidget::zoomIn(uint64_t _time)
{
        int64_t timeSpan = (m_maxTime - m_minTime) / 2;
        int64_t middle = (m_maxTime + m_minTime) / 2;
        if (_time != UINT64_MAX)
        {
                middle = _time;
        }

        int64_t newMinTime = middle - timeSpan * 2/3;
        int64_t newMaxTime = middle + timeSpan * 2/3;
```

```
        int64_t mintime = m_context-
>m_capture->getMinTime();
        int64_t maxtime = m_context-
>m_capture->getMaxTime();
        if (newMaxTime > maxtime)
        {
                uint64_t delta =
newMaxTime - maxtime;
                newMinTime -= delta;
        }

        if (newMinTime < mintime)
        {
                uint64_t delta =
mintime - newMinTime;
                newMaxTime += delta;
        }

        newMaxTime =
qMin(newMaxTime, (int64_t)m_context-
>m_capture->getMaxTime());
        newMinTime =
qMax(newMinTime, (int64_t)m_context-
>m_capture->getMinTime());

        animateRange(newMinTime,newM
axTime);
        m_actionZoomReset-
>setEnabled(true);
}

void GraphWidget::zoomOut()
{
        zoomOut(UINT64_MAX);
}

void GraphWidget::zoomOut(uint64_t
_time)
{
        int64_t timeSpan =
(m_maxTime - m_minTime) / 2;
        int64_t middle   =
(m_maxTime + m_minTime) / 2;
        if (_time != UINT64_MAX)
        {
                middle = _time;
        }

        int64_t newMinTime = middle
- timeSpan * 3/2;
        int64_t newMaxTime = middle
+ timeSpan * 3/2;

        int64_t mintime = m_context-
>m_capture->getMinTime();
        int64_t maxtime = m_context-
>m_capture->getMaxTime();
        if (newMaxTime > maxtime)
        {
                int64_t delta =
newMaxTime - maxtime;
                newMinTime -= delta;
        }

        if (newMinTime < mintime)
        {
                int64_t delta =
mintime - newMinTime;
                newMaxTime += delta;
        }

        newMaxTime =
qMin(newMaxTime, (int64_t)m_context-
>m_capture->getMaxTime());
        newMinTime =
qMax(newMinTime, (int64_t)m_context-
>m_capture->getMinTime());

        if ((newMinTime ==
(int64_t)m_context->m_capture-
>getMinTime()) && (newMaxTime ==
(int64_t)m_context->m_capture-
>getMaxTime()))
                m_actionZoomReset-
>setEnabled(false);

        animateRange(newMinTime,newM
axTime);
}

void GraphWidget::zoomReset()
{
        animateRange(m_context-
>m_capture->getMinTime(), m_context-
>m_capture->getMaxTime());
        m_actionZoomReset-
>setEnabled(false);
}

void GraphWidget::zoomSelect()
{
        animateRange(m_context-
>m_capture->getSnapshotTimeMin(),
m_context->m_capture-
>getSnapshotTimeMax());
        m_actionZoomReset-
>setEnabled(true);
}

void GraphWidget::zoomAnimEvent()
{
        invalidateScene();
}

void GraphWidget::markerSnapTo()
{
        uint64_t f =
m_hoverMarkerTime;
        uint64_t maxS = m_context-
>m_capture->getSnapshotTimeMax();
        uint64_t minS = m_context-
>m_capture->getSnapshotTimeMin();

        if (f>maxS)
                maxS = f;
        else
```

```
                if (f<minS)
                        minS = f;
                else
                {
                        // who is
closer?
                        uint64_t max
= maxS - f;
                        uint64_t min
= f - minS;
                        if (max <
min)
                                maxS
= f;
                        else
                                minS
= f;
                }

        selectFromTimes(minS,maxS);
}

void GraphWidget::markerSelectFrom()
{
        m_markerSelectFromTime =
m_hoverMarkerTime;

        if ((m_markerSelectFromTime
!= UINT64_MAX) &&
(m_markerSelectToTime !=
UINT64_MAX))

        selectFromTimes(m_markerSele
ctFromTime,m_markerSelectToTime);
        else
                invalidateScene();
}

void GraphWidget::markerSelectTo()
{
        m_markerSelectToTime =
m_hoverMarkerTime;

        if ((m_markerSelectFromTime
!= UINT64_MAX) &&
(m_markerSelectToTime !=
UINT64_MAX))

        selectFromTimes(m_markerSele
ctFromTime,m_markerSelectToTime);
        else
                invalidateScene();
}

void
GraphWidget::resizeEvent(QResizeEven
t* _event)
{
        RTM_UNUSED(_event);
        QSize newSize = size();

        m_scene->setSceneRect(-
newSize.width()/2, -
newSize.height()/2, newSize.width(),
newSize.height());

        if (m_grid)
                m_grid-
>parentResized();

        Q_FOREACH( GraphCurve* it,
m_curves ) {
                it->parentResized();
        }

        fitInView(m_scene-
>sceneRect());
        invalidateScene();
}

void
GraphWidget::wheelEvent(QWheelEvent*
_event)
{
        RTM_UNUSED(_event);
        int delta = _event->delta();
        QPoint position = _event-
>pos();
        uint64_t relTime =
mapPosToTime(position.x());
        if (delta < 0)
                zoomOut(relTime);
        else
                zoomIn(relTime);
}

void
GraphWidget::mousePressEvent(QMouseE
vent* _event)
{
        if (!m_context)
                return;

        QRect drawRect =
getDrawRect();
        QPointF pt =
mapToScene(_event->pos());

        if ((_event->buttons() &
Qt::LeftButton) &&
drawRect.contains(pt.x(),pt.y()))
        {
                m_LButtonDown   =
true;
                m_dragStartPos =
_event->pos();
        }

        if ((_event->buttons() &
Qt::RightButton) &&
drawRect.contains(pt.x(),pt.y()))
        {
                m_RButtonDown   =
true;
                m_dragStartPos =
_event->pos();
```

```
        }

        QGraphicsView::mousePressEve
nt(_event);
}

void
GraphWidget::mouseMoveEvent(QMouseEv
ent* _event)
{
        mouseMovement(_event->pos(),
_event->buttons());
        QGraphicsView::mouseMoveEven
t(_event);
}

extern QString getTimeString(float
_time, uint64_t* _mSec = NULL);

void
GraphWidget::mouseMovement(const
QPoint& _position, Qt::MouseButtons
_buttons)
{
        if (!m_context)
                return;

        QPoint gpt =
mapToGlobal(_position);
        QPointF spt =
mapToScene(_position);

        bool dontHideToolTip =
false;
        const
QVector<MarkerToolTip>&
        toolTips = m_markers-
>getTooltips();
        for (int i=0;
i<toolTips.size(); ++i)
        {
                if
(toolTips[i].m_rect.contains(spt) &&
!m_inContextMenu)
                {
                        QString
text(toolTips[i].m_text);
                        text += "\n"
+ tr("Time") + ": " +
QString::number(m_context-
>m_capture-
>getFloatTime(toolTips[i].m_time),'f
');
                        text += "\n"
+ tr("Thread") + ": 0x" +
QString::number(toolTips[i].m_thread
ID, 16);

        QToolTip::showText(gpt,
text, this, QRect(_position,
_position));

        dontHideToolTip = true;
                        m_hoverMarkerTime =
toolTips[i].m_time;
                        break;
                }
        }

        if
(rect().contains(_position) &&
m_LButtonDown && !m_inContextMenu)
        {
                uint64_t pL =
mapPosToTime(m_dragStartPos.x());
                uint64_t pR =
mapPosToTime(_position.x());
                uint64_t startTime =
qMin(pL,pR);
                uint64_t endTime =
qMax(pL,pR);

                rtm::GraphEntry
entry;
                m_context-
>m_capture->getGraphAtTime(endTime,
entry);

                float startTimeF =
m_context->m_capture-
>getFloatTime(startTime);
                float endTimeF =
m_context->m_capture-
>getFloatTime(endTime);

                QString ttip =
        tr("Start time") + ": " +
getTimeString(startTimeF) + "\n" +
tr("End time") + ": " +
getTimeString(endTimeF) + "\n" +



        QToolTip::showText(gpt,ttip,
this,QRect(_position, _position));
        }
        else
        {
                if (dontHideToolTip
== false)

        QToolTip::hideText();
        }

        if ((_buttons &
Qt::LeftButton) && m_LButtonDown &&
!m_isDragging)
                m_isDragging = true;

        if ((_buttons &
Qt::RightButton) && m_RButtonDown &&
!m_isPanning && m_actionZoomReset-
>isEnabled())
        {
                m_isPanning = true;
```

```
        setCursor(Qt::ClosedHandCurs
or);
        }

        if ((_buttons &
Qt::LeftButton) && m_isDragging)
        {
                uint64_t pL =
mapPosToTime(m_dragStartPos.x());
                uint64_t pR =
mapPosToTime(_position.x());
                m_select-
>setSelectRange(pL,pR);

                if (!m_context-
>m_capture->getFilteringEnabled())
                {
                        uint64_t
startTime = qMin(pL,pR);
                        uint64_t
endTime = qMax(pL,pR);
                        m_context-
>m_capture-
>setSnapshot(startTime,endTime);
                        emit
snapshotSelected();
                }
        }

        if ((_buttons &
Qt::RightButton) && m_isPanning)
        {

        setCursor(Qt::ClosedHandCurs
or);
                uint64_t pL =
mapPosToTime(m_dragStartPos.x());
                uint64_t pR =
mapPosToTime(_position.x());
                int64_t delta = pL -
pR;

                if (delta != 0)
                {

        m_dragStartPos = _position;
                        int64_t
newMinTime = m_minTime + delta;
                        int64_t
newMaxTime = m_maxTime + delta;

                        int64_t
mintime = m_context->m_capture-
>getMinTime();
                        int64_t
maxtime = m_context->m_capture-
>getMaxTime();
                        if
(newMaxTime > maxtime)
                        {
        uint64_t deltaU = newMaxTime
- maxtime;

        newMinTime -= deltaU;
                        }

                        if
(newMinTime < mintime)
                        {

        uint64_t deltaU = mintime -
newMinTime;

        newMaxTime += deltaU;
                        }

                        newMaxTime =
qMin(newMaxTime, (int64_t)m_context-
>m_capture->getMaxTime());
                        newMinTime =
qMax(newMinTime, (int64_t)m_context-
>m_capture->getMinTime());

                        m_minTime =
newMinTime;
                        m_maxTime =
newMaxTime;

                        emit
minMaxChanged();
                }
        }

        invalidateScene();
}

uint64_t GraphWidget::currentPos()
{
        QRect drawRect =
getDrawRect();
        QPoint gpt = QCursor::pos();

        QPoint position =
mapFromGlobal(gpt);
        QPointF pt =
mapToScene(position);

        if
(drawRect.contains(pt.x(),pt.y()) &&
!m_LButtonDown && !m_inContextMenu)
        {
                uint64_t pL =
mapPosToTime(position.x());
                float timeF =
m_context->m_capture-
>getFloatTime(pL);
                rtm::GraphEntry
entry;
                m_context-
>m_capture->getGraphAtTime(pL,
entry);
```

```
                QString timeStr =
getTimeString(timeF);

                QString ttip =
        tr("Time") + ": " + timeStr
+ "\n" +



        QToolTip::showText(gpt,ttip,
this,QRect(position, position));
                return pL;
        }

        return 0;
}

void
GraphWidget::mouseReleaseEvent(QMous
eEvent* _event)
{
        if (!m_context)
                return;

        if ((_event->button() ==
Qt::LeftButton) && m_LButtonDown)
                m_LButtonDown =
false;

        if ((_event->button() ==
Qt::RightButton) && m_RButtonDown)
        {
                m_RButtonDown =
false;

        setCursor(Qt::ArrowCursor);
        }

        if (_event->button() ==
Qt::LeftButton)
        {
                if (m_isDragging)
                {
                        uint64_t
time1 = mapPosToTime(_event-
>pos().x());
                        uint64_t
time2 =
mapPosToTime(m_dragStartPos.x());
                        if
(m_context->m_capture-
>getFilteringEnabled())
                        {

        uint64_t startTime =
qMin(time1,time2);

        uint64_t endTime =
qMax(time1,time2);

        m_context->m_capture-
>setSnapshot(startTime,endTime);
                                emit
snapshotSelected();
                        }

        m_actionZoomToSelection-
>setEnabled(true);
    tr("Usage") + ": " + m_locale.toString(
    tr("Time
r->setEnabled(true);
                        m_isDragging
= false;
                }
                else
                {
                        m_context-
>m_capture->setSnapshot(m_context-
>m_capture->getMinTime(), m_context-
>m_capture->getMaxTime());
                        m_select-
>setSelectRange(0,0);

        m_actionZoomToSelection-
>setEnabled(false);

        m_actionSnapSelectionToMarke
r->setEnabled(false);

        invalidateScene();
                        emit
snapshotSelected();
                }


        m_markerSelectFromTime =
(uint64_t)-1;
                m_markerSelectToTime    = (ui
        }

        QGraphicsView::mouseReleaseE
vent(_event);
}

void
GraphWidget::contextMenuEvent(QConte
xtMenuEvent* _event)
{
        if (!m_context)
                return;

        if (m_isPanning)
        {
                m_isPanning = false;
                return;
        }

        bool isInMarker = false;
        QPoint pt = QCursor::pos();
        QPointF spt =
mapToScene(mapFromGlobal(pt));
        const
QVector<MarkerToolTip>&
        toolTips = m_markers-
>getTooltips();
```

```
        for (int i=0;
i<toolTips.size(); ++i)
        {
                if
(toolTips[i].m_rect.contains(spt))
                {
                        isInMarker =
true;

        m_hoverMarkerTime =
toolTips[i].m_time;
                        break;
                }
        }

        m_inContextMenu = true;

        if (isInMarker)
                m_contextMenuMarker-
>exec(mapToGlobal(_event->pos()));
        else
                m_contextMenu-
>exec(mapToGlobal(_event->pos()));
}

void
GraphWidget::createCustomContextMenu
()
{
        m_actionZoomToSelection =
new QAction(QString(tr("Zoom to
selection")),this);
        m_actionZoomReset = new
QAction(QString(tr("Reset
zoom")),this);

        m_actionZoomToSelection-
>setEnabled(false);
        m_actionZoomReset-
>setEnabled(false);

        m_contextMenu = new QMenu();
        m_contextMenu-
>addAction(m_actionZoomToSelection);
        m_contextMenu-
>addAction(m_actionZoomReset);
        connect(m_contextMenu,
SIGNAL(aboutToHide()), this,
SLOT(showToolTip()));
        connect(m_actionZoomToSelect
ion, SIGNAL(triggered()), this,
SLOT(zoomSelect()));
        connect(m_actionZoomReset,
SIGNAL(triggered()), this,
SLOT(zoomReset()));

        m_actionSnapSelectionToMarke
r = new QAction(QString(tr("Snap
selection to marker")), this);
        m_actionSelectFromMarker =
new QAction(QString(tr("Select from
marker")), this);
        m_actionSelectToMarker = new
QAction(QString(tr("Select to
marker")), this);
        m_contextMenuMarker = new
QMenu();
        connect(m_contextMenuMarker,
SIGNAL(aboutToHide()), this,
SLOT(showToolTip()));

        m_actionSnapSelectionToMarke
r->setEnabled(false);

        m_contextMenuMarker-
>addAction(m_actionSnapSelectionToMa
rker);
        m_contextMenuMarker-
>addAction(m_actionSelectFromMarker)
;
        m_contextMenuMarker-
>addAction(m_actionSelectToMarker);

        connect(m_actionSnapSelectio
nToMarker, SIGNAL(triggered()),
this, SLOT(markerSnapTo()));
        connect(m_actionSelectFromMa
rker, SIGNAL(triggered()), this,
SLOT(markerSelectFrom()));
        connect(m_actionSelectToMark
er, SIGNAL(triggered()), this,
SLOT(markerSelectTo()));
}

void GraphWidget::showToolTip()
{
        m_inContextMenu = false;
}
```