Київський політехнічний інститут імені Ігоря Сікорського

Кафедра АПЕПС

**Алгоритмізація та програмування – 2: Процедурне програмування**

**КУРСОВА РОБОТА**

«**Калькулятор**»

*(ТЕМА)*

Дата «\_\_\_» \_\_\_\_\_\_\_ 2022 Виконав: студент \_1\_ курсу

гр. ТР-15

Руденко Владислав Ігорович

*(П.І.Б.)*

Оцінка «\_\_\_\_\_\_\_\_\_» Перевірив: \_\_Крячок\_\_\_\_\_

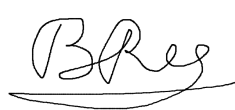
Дата «\_\_\_» \_\_\_\_\_\_\_ 2022 \_Олександр Степанович\_\_

*(П.І.Б., підпис)*

Київ – 2022

КАЛЕНДАРНИЙ ПЛАН

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Найменування етапів курсової роботи | Строк виконання етапів роботи | Примітки |
|  | Отримання завдання до КР | 9.02.2022 |  |
|  | Створення ідеї алгоритму | 15.02.2022 - 18.02.2022 |  |
|  | Підготовка презентації | 21.02.2022 - 25.02.2022 |  |
|  | Презентація КР (опис цілі роботи та засобів її розробки) | 01.03.2022 - 15.03.2022 |  |
|  | Створення програми | 16.03.2022 - 25.04.2022 |  |
|  | Здача програмного продукту | 26.04.2022 - 29.04.2022 |  |
|  | Перевірка КР на плагіат | 17.05.2022 - 19.05.2022 |  |
|  | Передача КР в АРХІВ КПІ | 20.05.2022 |  |
|  | ЗАХИСТ Курсової Роботи | 23.05.2022 - 28.05.2022 |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |



Студент: \_\_\_\_\_\_\_\_\_\_\_\_ Руденко. В.І

(підпис)

Керівник роботи: \_\_\_\_\_\_\_\_\_\_\_\_ Крячок О.С

  (підпис)

**Анотація**

Результат роботи являє собою програмний додаток під назвою “Калькулятор”. Який спроможний проводити складні математичні операції з використанням базових тригонометричних функцій та проводити конвертування системи числень одного виду в інший. Проводиться аналіз різних методів реалізування поставленої задачі. Розроблено алгоритм опрацювання різних задач та переглянуто можливість використання данної програми спільнотою. Для реалізації задачі було використано QT Creator - інтегроване середовище розробки, яке призначене для створення додатків. Середовище має на своїй базі різного виду бібліотеки, що дають змогу користувау створити лояльний для оператору інтерфейс.

Під час розробки було використано мову програмування С++, а також для деталізації та покращення інтерфейсу взаємодії використано мову html. Під час роботи було використано власний алгоритм опрацювання задачі.

**Annotation**

The result of the work is a program application called “Calculator”.

It is able to carry out complex mathematical operations using basic

trigonometric functions and convert the number system into the different types.

It analyzes different methods of tasks. The algorithm of processing the

different tasks has been developed and the possibility of using this program

by the community has been reviewed. QT Creator was being used to accomplish this task .it is an integrated development environment designed for creating applications. The environment has different kinds of libraries on its base, which

allows the user to create an operator-friendly interface.

The programming language C was used during the development,as well as the

Html language was used for detalization and improving the interface of interaction. The own algorithm of problem processing was used during the work

**Технічне завдання**

Ціллю роботи являється створення програми, що здатна за допомогою стандартних наборів математичних, тригонометричних функцій і функцій перетворення систем числення обчислювати задані користувачем значення, аналогічна назва такої програми - калькулятор.

Першими пристосуваннями для обчислень були, ймовірно, лічильні палички, які й сьогодні використовуються в початкових класах багатьох шкіл для навчання лічбі. Розвиваючись, ці пристосування ставали складнішими, так в 1643 був створенний та в подальшому суттєво вдосконалений перший калькулятор. НА сьогоднішній день калькулятори мають електронну форму та інтегровані майже в кожен комп’ютер та інший обчислювальний пристрій.

Робота включає декілька ключових етапів виконання:

1. Створення інтерфейсу взаємодії з користувачем.
2. Створення методів обробки наданих користувачу полів (кнопки операцій, екран виводу, тощо).
3. Реалізація можливості зміни поточного вікна зі зміною його функціоналу.

**Зміст**

Анотація (3)

Технічне завдання (5)

Вступ (7)

1. Калькулятор (8)

1.1 Поняття системи числення (9)

1.2 Програмне середовище QT (11)

2. Опис програмної реалізації (12)

3. Опис отриманих результатів (20)

4. Висновок (22)

5. Список використаних джерел (23)

6. Додаток (24)

**Вступ**

Сучасні калькулятори являються дуже поширеними і включені в більшість обчислювальних приладів, вони використовуються у всіх сферах діяльності, адже допомагають людям виконувати їхню роботу швидше та ефективніше. В ході еволюції калькуляторів, вони доповнювались різними методами та функціями. Татим чином сьогодні існують різні варіації звичайного калькулятору: від найпростішого (який здатен використовувати базові математичні функції) до калькуляторів погоди, траекторії руху, інженерні та інші). Калькулятори маєть змогу не тільки проводити певні обчислення за заданими формулами, а й візуалізовувати, в деяких ситуаціях, свій результат (графіки, схеми, тощо), що суттєво допоможить при вирішенні певного роду задач.

Метою роботи є інженерний калькулятор, який модифікований модулем конвертацій систем числення. Такий модуль повинен буде використовуючі власні методи опрацьовувати та конвертувати значення однієї системи числення в іншу. Такий пристрій зможе стати в нагоді при обчисленні складних функцій або переводу значень однієї системи числення в іншу.

**1.Калькулятор**

Калькулятор - електронно-обчислювальний пристрій для проводження операцій над значеннями або алгеброїчними формулами. Кожен сучасний калькулятор має змогу опрацьовувати значення з використанням знаків після кому, що дозволяє зробити обчислення максимально точними.

Сучасні калькулятори поділяються на такі базові види:

1. Простий - використовуються для обчислень з використанням базових математичних функцій (такі як додавання, віднімання, множення, ділення). Ці пристрої дуже легкі в своїй конструкції, але на сьогоднішній день вони рідко зустрічаються, адже були витіснені інженерними калькуляторами.
2. Інженерні - пристрої повишеної складності, які призначене для складних обрахунів з використанням тригонометричних та інших методів, зачасту такі калькулятори використовуються інженерами, науковцями різних галузей, працівниками професій де необхідні пошлиблені знання математики та школярів.
3. Фінансові - ристрої направлені на роботу в сферах фінінсової діяльності людини, такі калькулятори позбавлені лишніми для них функціями, але включають в себе можливість конвертації різних валют та опрацювання їх значеннь.
4. Графічні - спецефічний вид калькулятора, на перший погляд який нагадує інженерний калькулятор, але допомагає візуалізувати результат обчислень. Він спроможній не тільки обчислювати певні значення, а ще й на їх основі опрацьовувати певного роду графіки, фігури та інші елементи графіки.

Отже калькулятор це незаміннний в сучасному представленню світу. Він має багато варіацій, що робить цей пристрій універсальний. Складність та функціонал якого обмежується тільки потребами та фантазією розробника. Винахід калькулятора значимо сприяв прискоренню усіх процесів нашого часу, коли розрахунки є дуже складовою усіх сфер діяльності.

**1.2 Поняття системи числення**

Система числення - сукупність правил і знаків, які допомагають відобразити(кодувати) будь-яке значення(яке не є відємним). Такі системи мають певні вимоги для коректної своєї роботи, серед них найважливішими є вимоги однозначного кодування (0..1..) з деякої скінченної множини(діапазону P) та скінченне число кроків і можливості виконання арифметичних і логічних операцій). Системи числення розв’язують задачу ефективного переходу від зображених числел та номерів, які мають мінімальну кількість цифр. Сама ефективність залежить від вдалості вибору системи числення. Розрізняють такі типи систем числення:

1. Позиційні - де використовуються ідентичні цифри та результатом стає різні значення, які залежать від позиції. Таким чином саме позиціє впливає на значення, яке буде отримано. Основною системи числення є вага кожної позиції, яка кратна деякому натуральному числу (n>1). Іншими словами основа - кількість використаних при записі числа символів. За допомогою варіацій пар чисел від двох до десяти, в різній комбінаціях, можна зобразити будь-яке дійсне число.
2. Змінаша - є узагальненою системою счислення основою якоє є b вона часто буває відношена до позиційних систем числення. Основою такої системи числення є послідовність деяких числен, що зростає за певною формулою () і кожне число представляється як лінійна комбінація.
3. Непозиційна система - на відмінно від попередніх систем значення не залежить від позиції у числі. При цьому система має обмеження на саму позицію цифр, наприклад їх розташування по спаданню чи у згрупованні за значенням. Незважаючи на це, ці умови не є принциповими для розуміння записаних такими системами чисел. Прикладом є римська система числення, в якій використовуються латинські букви (I(1), II(2), III(3), IV(4),V(5)…X(9)…та інші).

У нумізматиці особлива увага приділяється десятковій, дванадцятковій, четвертній та шістковій системі. Інформаційні ж технології білше звертають увагу на двійкову, десяткову, вісімкову та шістнадцяткову системи.

Таким чином поняття систем счислення являється актуальним, адже використовує різні свої варіації в різних сфера діяльності. Розуміння алгоритму роботи системи числення є необхідною складовою для виконання проекту.

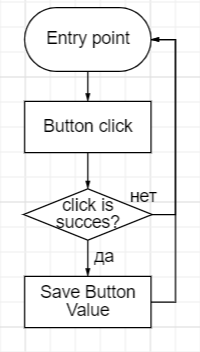
**1.2 Програмне середовище QT**

Інтегроване середовище розробки QT Creator призначене для створення різного роду застосунків з використанням бібліотек QT. Основною мовою програмування є С++, але середовище підтримує і інші популярні мови програмування. Для реалізації сценаріїв роботи використовуюється мова JavaScript. А структури і елементи інтерфейсу задаються CSS-подібними блоками. QT creator має змогу використовувати GCC або Microsoft Visual Studio компілятори. Для платформи Windows (на якій ведеться розробка проекту) бібліотека компілюється компілятором MinGW. Програмне середовище має можливість подальшого представлення проекту у вигляді окремого додатку, що робить його більш універсальним та гнучким.

1. **Опис програмної реалізації**

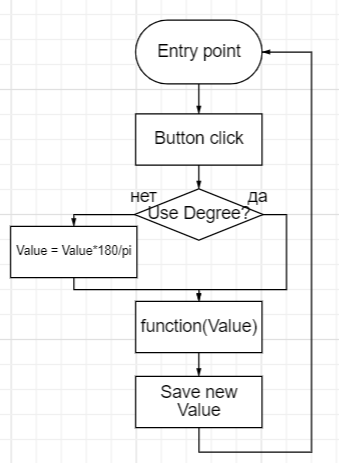
Інтегроване середовище QT дало змогу спроєктувати графічний інтерфейс програми (GUI) для покращення взаємодії з користувачем. Спочатку було створенно робоче поле, на яком, в майбутньому, було розміщено усі необхідні елементи взаємоді, такі як кнопки операцій, кнопки, які відповідають за значення, поле для виводу інформації, та меню вибору поточнного середовища. Кожна з цих елементів має свій алгоритм подій.

Прикладом такого алгоритму є одна з кнопок вводу значень. Він є найпростішим в данній програмі, адже лише перевіряє чи була натиснута кнопка, якщо так - її результат записується на екран користувачеві та в внутрішній буфер.



Блок-Схема 1 (Найпростіший алгоритм події кнопки значення)

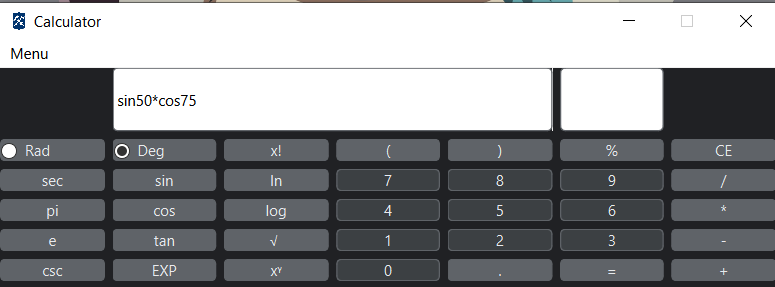
Таким чином майже кожна кнопка має схожий алгоритм подій, їхня різниця тільки в тому, що буде зберігати цей алгоритм, чи то значення, чи то операція.

Але при тому кнопки виконання тригонометричних функцій мають додатково ще пункт виконання певної операції. Так кнопка, яка відповідає за знаходження синусу певного числа, вираховує чи користувач бажає провести обчислення з використанням градусів, чи необхідно перевести це значення в радіани, після чого відбувається сам обрахунок синусу даного значення.  
  


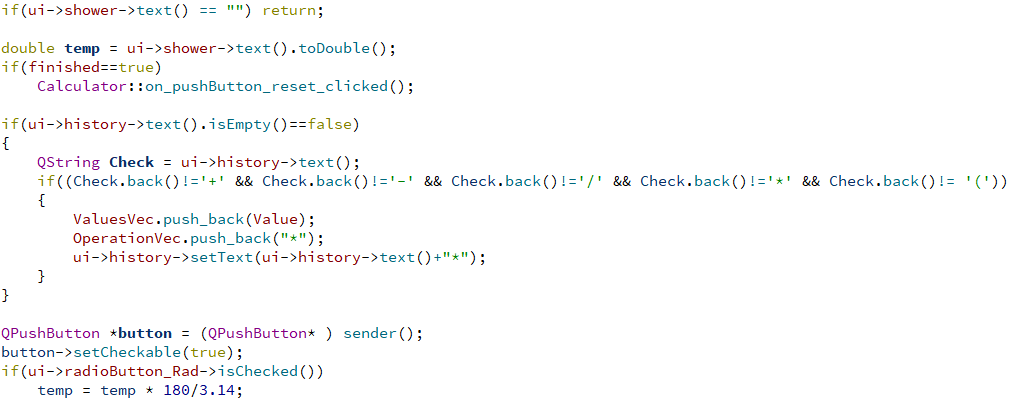
Блок-Схема 2 (Алгоритм тригонометричної функції)

Таким чином як тільки програма була запущена користувач має змогу відразу ж почати виконання. Алгоритм передбачає алгебраїчну логіку виконання, що дозволяє користувачеві використовувати повний спектор можливостей сучасного калькулятора (використання дужок, пріорітетних операцій, тощо). Але алгоритм має свої спецефичні можливості, так для обчислення будь-якої функції треба спочатку ввести значення, а потім вибрати операцію, яка буде проведена над цим значенням, так було введено ще декілька правил для обчислення, наприклад: якщо користувач декілька раз поспіль намагається знайти значення тригонометричної функції від деякого числа, не додаючи проміжні знаки операцій (+,-,\*,/), програма, провівши певну перевірку,

Додає між ними операцію множення, що є стандартною практикою в алгеброїчних моделях задач. Також алгоритм передбачає неможливість виконання деякої функції, якщо поле для вводу залишається порожнім.



Приклад 1 (ситуація з використанням функцій підряд)

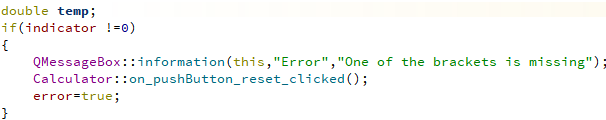


Додаток 1 (Перевірка стану введення деякої функці)

Як бачимо, алгоритм виконує цю ситуацію так, як це б сприймала людина. Також, хоч алгоритм і виводить користувачеві функцію з початковим значенням, він проводить обрахунки не чекаючі завершення всього вводу, оскільки тригонометричні та деякі інші функції мають найвищий пріорітет виконання.

Завершення ж вводу та початок основної цепочки обрахунків починається тільки після натискання користувачем кнопки дорівнює (=). Програма відразу завершує запис з урахуванням останнього введеного значення та перевіркою останнього введеного знака операцій (якщо він пристуній і йому не вистачає значення - він виключається з черги виконання), після чого проводить ряд базових, для себе операцій.

По-перше, це підрахунок кількості наявних дужок, та правильність їх розташування (кількість відкриваючих дужок повинна дорівнювати кількості закриваючих, закриваюча дужка повинна бути тільки пілся відкриваючої). При успішному проходження перевірки алгоритм виставляє ліміт обчислення, який дорівнює краям даних дужок, таким чином спочатку будуть виконуватись операції в “найвищій точці”.



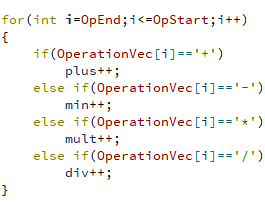
Додаток 2 (Алгоритм звірення кількості дужок)



Додаток 3 (перевірка розташування дужок)

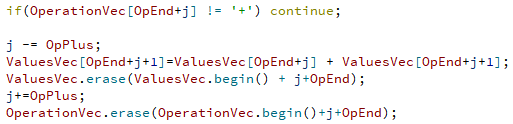
Якщо дужок не було знайдено, алгоритм продовжує свое виконання в стандартних умовах.

По-друге, відбувається внутрішній підрахунок кількості операцій в середині границь (якщо дужки відсутні - лімітом є початок і завершення всіх операцій). Це зроблено для стабілізації виконання і скорочення витраченого часу на пошук окремих елементів в майбутньому.



Додаток 4 (Пошук кількості елементів в заданих границях)

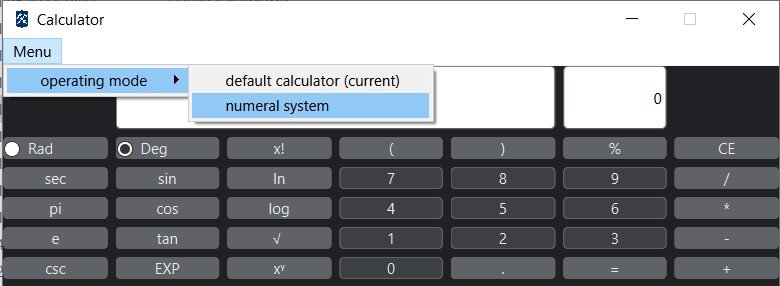
Кількість певних елементів використовується для поставлення порядку виконання операцій. Кожна операція, яка була проведена, замінює в буфері другий операнд на результат операції, пілся чого відбувається видалення першого операнду цієї операції та знак самого оператору. Таким чимном другий оеранд даної операції ставє доступний для використання в інших виразах.



Додаток 5 (Опрацювання певного виразу з додаванням)

Ці операції відбуваються поки не залишиться ні одного знаку оператора та один результут в буфері програми.

Після завершення виконання програма позначає сама для себе що виконання завершенно, та при наступному нажатті будь якої кнопки алгоритм автоматично застосовує функцію кнопки (CE) яка очищує поля вводу та внутрішні буфера обміну. Що дозволяє алгоритму почати виконання знову.



Приклад 2 (переведення в режим конвертора)

Сама програма крім звичайного калькулятора передбачає конвертування сисчтем числень, для початку користування даним конвертором необхідно зайти до меню програми, та обрати необхідний пункт.

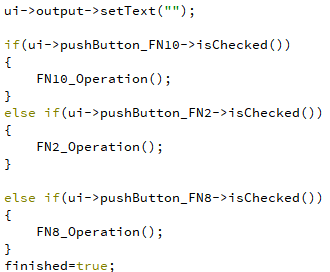
Інтерфейс конвертора схожий з інтерфейсом калькулятора тільки дизайном вікна. Замість привичного калькулятору функціоналу, конвертор має дещо урізані можливості, які йому не потрібні. Таким чином конвертор має тільки кнопки цифр, дорівнює та кнопку CE для очистки виконання. Натомість конвертор має кнопки для вибору режимів конвертування, усього їх 7 (3 на вхід, 3 на вихід, ситуаціїколи необхідно використати одну й ту саму систему конвертації не враховуються). Для вхідних було обрано двійкову, десяткову та вісімкову систему числення. Для виводу пропонується двійкова, десяткова та шістнадцяткова. Шістнадцяткова система вводу була замінена на вісімкову, хоча використання шістнадцяткової системи полегшело виконання алгоритму, але ставило користувача в більш скрутні для розуміння умови, оскільки систем числення після десяткової використовуються крім значень букви, що збільшило б кількість кнопок та розмір інтерфейсу, через що він би втратив свою компактність.

Алгоритм вводу значень не відрізняється від алгоритму вводу для калькулятора, але має свою додаткову умову. Якщо обрана конвертація із двійкової системи, користувачу перекривається доступ до усіх кнопок шляхом їх відключення, оскільки двійкова система передбачає використання всього лише двох значень (0 та 1).

Пілся введення певного значення та обрання систем конвертації натискається кнопка дорівнює (=). Вона викликає метод, який проводить певні первинні перевірки, після чого проводить необхідні операції.

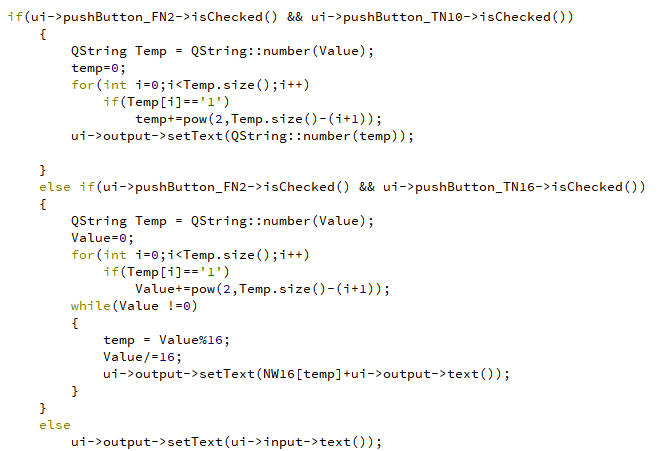
Однією з перевірок є наявність значень в полі для вводу (якщо відсутні - операції не проводяться) та правильність вибору користувачем систем счислень (можливий варіант коли користувач проігнорував вибір цих пунктів) в такому випадку алгоритм застосовує параметри за замовчуванням (перевід з десяткової до двійкової систем числення).

Для переходу назад до режуму калькулятора користувачу необхідно повторно вибрати меню і там обрати режим калькулятора.



Додаток 6 (Перевірка вибору конвертацій)

Процес опрацювання відбувається за допомогою перевірки вибраного виду конвертації, якщо варіант співпадає - викликається певна функція, яка проводить операції з данною конвертацією.

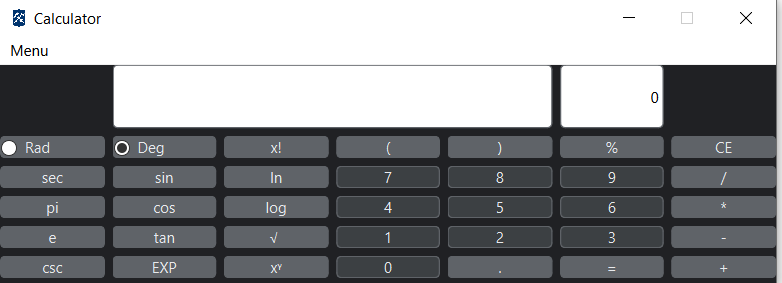


Додаток 7 (приклад операції конвертування)

Після завершення опрацювання алгоритм позначає про закінчення виконання та вивід результатів на екран. При повторному натиску будь-яких кнопок відбувається виклик методу CE, який дозволяє почати виконання знову.

**Опис отриманих результатів**

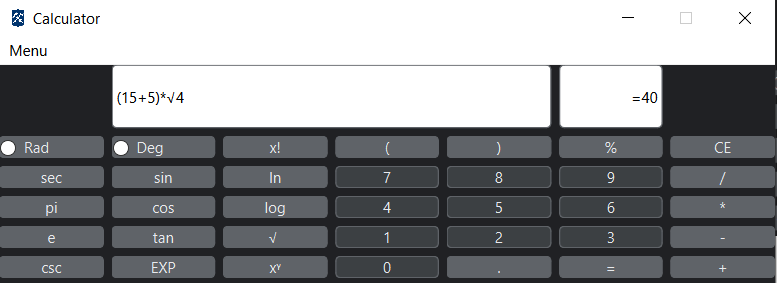
Так як калькулятор має два режими роботи, звичайний та конвертор:



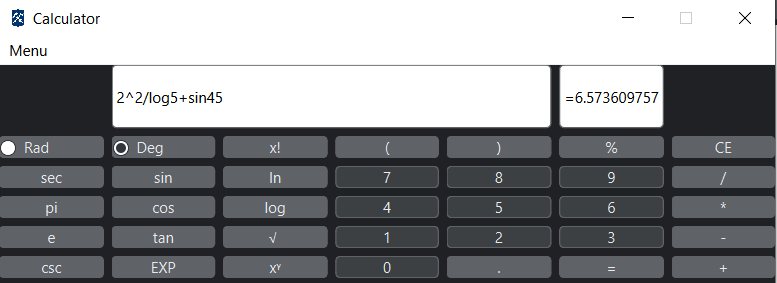
Приклад 3 (Стандартне вікно калькулятора)

Оскільки калькулятор розроблювався для використання в ньому складних операцій він не має скороченої форму, натомість він відразу є повноцінним, що забезпечує швидкісне виконання шляхом (відкрив->виконав).

Дизайн був взятий зі стандарного web калькулятора від google, оскільки він є строгим, інформативним та компактним, що робить його дуже зручним та приємним у користуванні.

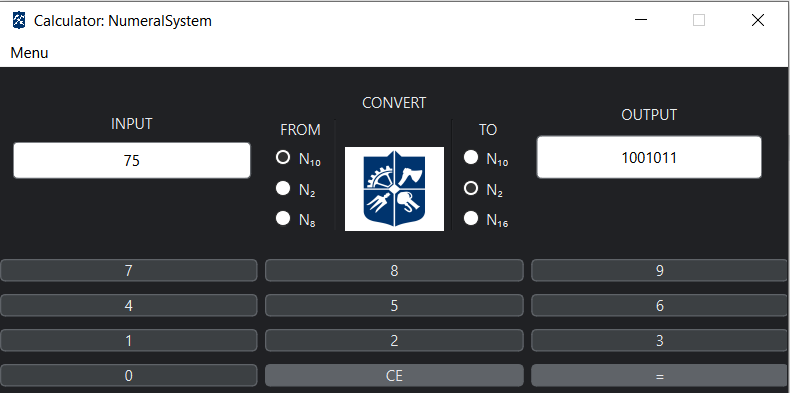


Приклад 4 (Опрацювання певного виразу)

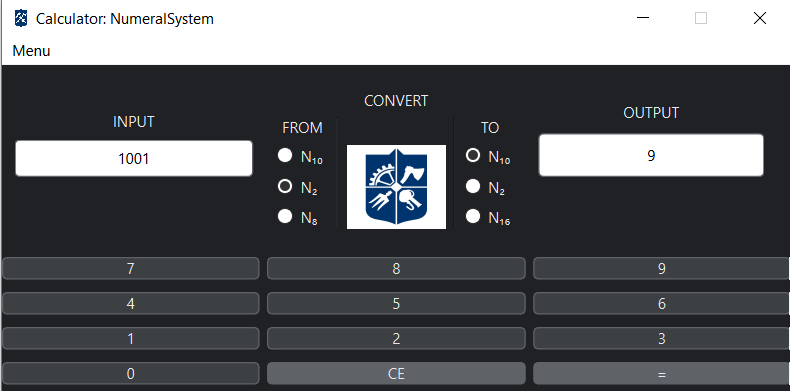


Приклад 5 (ще один приклад опрацювання калькулятору)

Другий режим калькулятора це конвертор, перехід відбувається за допомогою меню.



Приклад 6 (Опрацювання певного виразу конвертора)



Приклад 7 (ще один приклад застосування конвертора)

**Висновки**

Під час виконання курсової роботи “Калькулятор” було дослідженно методи взаємодії з користувачем, дослідженно різні алгоритми та методами калькулятора та конвертора, та його поведінку при певних умовах.

В результаті роботи розроблено програмний продукт, який здатен обчислювати вирази за допомогою математичних та тригонометричних функцій - тобто калькулятор, але який модифікований і здатний переводити системи числення.

Інтерфейс програми розроблено завдяки інтегрованому середовищу QT, бібліотеки якого дозволили гнучко та ефективно взаємодіяти з користувачем. Алгоритми програми написано мовою програмування С++, також використано елементи html для створення та редагування інтерфейсу. Алгоритм працює ефективно та коректно.  
 Для реалізаціє програмного продукту використовувались різні методи та кнопки, що їх викликали. Отримано навички створення різного типу додатків. Запропоновано методи взаємодії з користувачем та код, що здійснює різного виду обрахунки.

**СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ**

1. Калькулятор - Wikipedia.org

Режим доступу: https://uk.wikipedia.org/wiki/Калькулятор

1. Типи калькуляторів - RankRed.org

Режим доступу: https://www.rankred.com/different-types-of-calculators/

1. Інтегроване середовище Qt - Wilipedia.org

Режим доступу: https://uk.wikipedia.org/wiki/Qt\_Creator

1. Функціонал та методи користування QT - qt.io

Режим доступу: https://doc.qt.io/qtcreator/

1. Системи числення - Wilipedia.org

Режим доступу: https://uk.wikipedia.org/wiki/Система\_числення

1. Переведення систем числення - Google.com

Режим доступу: https://sites.google.com/site/sistemicislennaveronika/pervedenna-cisel-z-odniei-sistemi-cislenna-v-insu

1. Тригонометричні функції - Wilipedia.org

Режим доступу: https://uk.wikipedia.org/wiki/Тригонометрія

1. Мова програмування С - williamspublishing.com

Режим доступу: http://www.williamspublishing.com/Books/978-5-8459-0891-9.html#ogl

1. Концепція роботи калькулятора - YouTube.com

Ружим доступу: https://www.youtube.com/watch?v=Vk-tGND2bfc

1. Система числення в програмуванні - YouTube.com  
   Ружим доступу: https://www.youtube.com/watch?v=c5FSYMpsLN0

**Додаток**

**(частковий текст програми)**

**Calculator.cpp**

#include "calculator.h"

#include "./ui\_calculator.h"

#include <vector>

#include <math.h>

#include <QMessageBox>

#include "numeralsystem.h"

double **tempValue**;

using namespace std;

Calculator::**Calculator**(QWidget \***parent**) :

QMainWindow(*parent*),

ui(new Ui::Calculator)

{

ui->setupUi(this);

numeralsys = new NumeralSystem();

connect(numeralsys, &NumeralSystem::defoultcalculator, this, &Calculator::show);

connect(ui->pushButton\_0,SIGNAL(clicked()),this,SLOT(number()));

connect(ui->pushButton\_1,SIGNAL(clicked()),this,SLOT(number()));

connect(ui->pushButton\_2,SIGNAL(clicked()),this,SLOT(number()));

connect(ui->pushButton\_3,SIGNAL(clicked()),this,SLOT(number()));

connect(ui->pushButton\_4,SIGNAL(clicked()),this,SLOT(number()));

connect(ui->pushButton\_5,SIGNAL(clicked()),this,SLOT(number()));

connect(ui->pushButton\_six,SIGNAL(clicked()),this,SLOT(number()));

connect(ui->pushButton\_seven,SIGNAL(clicked()),this,SLOT(number()));

connect(ui->pushButton\_eight,SIGNAL(clicked()),this,SLOT(number()));

connect(ui->pushButton\_nine,SIGNAL(clicked()),this,SLOT(number()));

connect(ui->pushButton\_point,SIGNAL(clicked()),this,SLOT(numberWithPoint()));

connect(ui->pushButton\_plus,SIGNAL(clicked()),this,SLOT(operation()));

connect(ui->pushButton\_min,SIGNAL(clicked()),this,SLOT(operation()));

connect(ui->pushButton\_mult,SIGNAL(clicked()),this,SLOT(operation()));

connect(ui->pushButton\_div,SIGNAL(clicked()),this,SLOT(operation()));

connect(ui->pushButton\_cos,SIGNAL(clicked()),this,SLOT(TrigoNfunction()));

connect(ui->pushButton\_sin,SIGNAL(clicked()),this,SLOT(TrigoNfunction()));

connect(ui->pushButton\_tan,SIGNAL(clicked()),this,SLOT(TrigoNfunction()));

connect(ui->pushButton\_csc,SIGNAL(clicked()),this,SLOT(TrigoNfunction()));

connect(ui->pushButton\_sec,SIGNAL(clicked()),this,SLOT(TrigoNfunction()));

connect(ui->pushButton\_log,SIGNAL(clicked()),this,SLOT(Nfunction()));

connect(ui->pushButton\_In,SIGNAL(clicked()),this,SLOT(Nfunction()));

}

Calculator::~**Calculator**()

{

delete ui;

}

void Calculator::**number**()

{

if(finished==true)

{

Calculator::on\_pushButton\_reset\_clicked();

}

QPushButton \***button** = (QPushButton\* ) sender();

Value = ((ui->shower->text()+button->text()).toDouble());

QString **update**;

update = QString::number(Value);

ui->shower->setText(update);

}

void Calculator:: **numberWithPoint**()

{

if(ui->shower->text().contains('.') == false)

{

ui->shower->setText(ui->shower->text()+".");

}

}

void Calculator::**on\_pushButton\_per\_clicked**()

{

if(ui->shower->text() != "")

{

Value = Value\*0.01;

ui->history->setText(ui->history->text()+ui->shower->text()+"%");

ui->shower->setText("");

}

}

int **factorial**(int **i**)

{

if (i==0) return 1;

else return i\*factorial(i-1);

}

void Calculator::**on\_pushButton\_factorial\_clicked**()

{

if(ui->shower->text() != "")

{

Value = factorial((int)Value);

ui->history->setText(ui->history->text()+ui->shower->text()+"!");

ui->shower->setText("");

}

}

void Calculator::**Nfunction**()

{

if(finished==true)

Calculator::on\_pushButton\_reset\_clicked();

if(ui->history->text().isEmpty()==false)

{

QString **Check** = ui->history->text();

if((Check.back()!='+' && Check.back()!='-' && Check.back()!='/' && Check.back()!='\*'))

{

ValuesVec.push\_back(Value);

OperationVec.push\_back("\*");

ui->history->setText(ui->history->text()+"\*");

}

}

QPushButton \***button** = (QPushButton\* ) sender();

button->setCheckable(true);

if(ui->shower->text() == "") return;

if(ui->pushButton\_In->isCheckable())

{

ui->history->setText(ui->history->text()+"ln");

Value = log(ui->shower->text().toDouble());

}

else if(ui->pushButton\_log->isCheckable())

{

if(ui->shower->text().toDouble()==0)

{

QMessageBox::information(this,"Error","Unable to find logarithm of 0");

Calculator::on\_pushButton\_reset\_clicked();

error=true;

}

else

{

ui->history->setText(ui->history->text()+"log");

Value = log10(ui->shower->text().toDouble());

}

}

if(error!=true)

{

ui->history->setText(ui->history->text()+ui->shower->text());

ui->shower->setText("");

}

button->setCheckable(false);

}

void Calculator::**TrigoNfunction**()

{

if(ui->pushButton\_pow->isCheckable() == true)

{

}

if(ui->shower->text() == "") return;

double **temp** = ui->shower->text().toDouble();

if(finished==true)

Calculator::on\_pushButton\_reset\_clicked();

if(ui->history->text().isEmpty()==false)

{

QString **Check** = ui->history->text();

if((Check.back()!='+' && Check.back()!='-' && Check.back()!='/' && Check.back()!='\*' && Check.back()!= '('))

{

ValuesVec.push\_back(Value);

OperationVec.push\_back("\*");

ui->history->setText(ui->history->text()+"\*");

}

}

QPushButton \***button** = (QPushButton\* ) sender();

button->setCheckable(true);

if(ui->radioButton\_Rad->isChecked())

temp = temp \* 180/3.14;

if(ui->pushButton\_tan->isCheckable())

{

ui->history->setText(ui->history->text()+"tan");

Value = tan(temp);

}

else if(ui->pushButton\_cos->isCheckable())

{

ui->history->setText(ui->history->text()+"cos");

Value = cos(temp);

}

else if(ui->pushButton\_sin->isCheckable())

{

ui->history->setText(ui->history->text()+"sin");

Value = sin(temp);

}

else if(ui->pushButton\_sec->isCheckable())

{

ui->history->setText(ui->history->text()+"sec");

Value = 1/cos(temp);

}

else if(ui->pushButton\_csc->isCheckable())

{

ui->history->setText(ui->history->text()+"csc");

Value = 1/sin(temp);

}

ui->history->setText(ui->history->text()+ui->shower->text());

ui->shower->setText("");

button->setCheckable(false);

}

void Calculator::**on\_pushButton\_root\_clicked**()

{

if(ui->shower->text() == "") return;

ui->history->setText(ui->history->text()+"√"+ui->shower->text());

Value = sqrt(ui->shower->text().toDouble());

ui->shower->setText("");

}

void Calculator::**on\_pushButton\_pow\_clicked**()

{

if(ui->pushButton\_pow->isCheckable() == true)

{

ui->history->setText(ui->history->text()+ui->shower->text());

Value = pow(temp, Value);

ui->shower->setText("");

ui->pushButton\_pow->setCheckable(false);

}

else

{

temp = Value;

ui->history->setText(ui->history->text()+ui->shower->text()+"^");

ui->shower->setText("");

ui->pushButton\_pow->setCheckable(true);

}

}

void Calculator::**on\_pushButton\_pi\_clicked**()

{

Value = 3.1415;

QString **update**;

update = QString::number(Value);

ui->shower->setText(update);

}

void Calculator::**operation**()

{

if(ui->pushButton\_pow->isCheckable() == true)

{

Calculator::on\_pushButton\_pow\_clicked();

}

QPushButton \***button** = (QPushButton\* ) sender();

if(ui->history->text().isEmpty() == false &&

ui->shower->text().isEmpty())

{

QString **Check** = ui->history->text();

if((Check.back()=='+' || Check.back()=='-' || Check.back()=='/' || Check.back()=='\*'))

{

OperationVec.pop\_back();

Check.replace(Check.size()-1,1,button->text());

ui->history->setText(Check);

}

else

{

ui->history->setText(ui->history->text()+ui->shower->text()+button->text());

ValuesVec.push\_back(Value);

}

}

else

{

ui->history->setText(ui->history->text()+ui->shower->text()+button->text());

ValuesVec.push\_back(Value);

}

ui->shower->setText("");

OperationVec.push\_back(button->text());

button->setCheckable(false);

}

void Calculator::**on\_pushButton\_EXP\_clicked**()

{

if(ui->shower->text() != "")

{

ui->history->setText(ui->history->text()+"exp"+ui->shower->text());

Value = exp(ui->shower->text().toDouble());

ui->shower->setText("");

}

}

void Calculator::**on\_pushButton\_e\_clicked**()

{

Value = 2.718;

QString **update**;

update = QString::number(Value);

ui->shower->setText(update);

}

void Calculator::**on\_pushButton\_openbrace\_clicked**()

{

if(finished==true)

Calculator::on\_pushButton\_reset\_clicked();

indicator++;

ui->history->setText(ui->history->text()+"(");

OperationVec.push\_back("(");

NumberOfBrace++;

}

void Calculator::**on\_pushButton\_closebrace\_clicked**()

{

if(ui->pushButton\_pow->isCheckable() == true)

{

Calculator::on\_pushButton\_pow\_clicked();

}

if(finished==true)

Calculator::on\_pushButton\_reset\_clicked();

indicator--;

ui->history->setText(ui->history->text()+ui->shower->text()+")");

Value = ui->shower->text().toDouble();

ui->shower->setText("");

OperationVec.push\_back(")");

NumberOfBrace++;

}

void Calculator::**on\_pushButton\_reset\_clicked**()

{

ValuesVec.clear();

OperationVec.clear();

ui->history->setText("");

ui->shower->setText("0");

NumberOfBrace = indicator = 0;

finished=false;

error=false;

}

void Calculator::**on\_pushButton\_equ\_clicked**()

{

if(ui->pushButton\_pow->isCheckable() == true)

{

Calculator::on\_pushButton\_pow\_clicked();

}

QPushButton \***button** = (QPushButton\* ) sender();

if(ui->history->text().isEmpty()==false &&

ui->shower->text().isEmpty())

{

QString **Check** = ui->history->text();

if((Check.back()=='+' ||

Check.back()=='-' ||

Check.back()=='/' ||

Check.back()=='\*'))

{

OperationVec.pop\_back();

Check.replace(Check.size()-1,1,button->text());

ui->history->setText(Check);

}

else

{

ui->history->setText(ui->history->text()+ui->shower->text());

ValuesVec.push\_back(Value);

}

}

else

{

ui->history->setText(ui->history->text()+ui->shower->text());

ValuesVec.push\_back(Value);

}

ui->shower->setText("");

Calculator::equ\_solution();

if(!error)

{

ui->shower->setText("="+QString::number(ValuesVec[0], 'g', 10));

finished=true;

}

}

void Calculator::**equ\_solution\_res**()

{

int **min**=0,**plus**=0,**mult**=0,**div**=0;

for(int **i**=OpEnd;i<=OpStart;i++)

{

if(OperationVec[i]=='+')

plus++;

else if(OperationVec[i]=='-')

min++;

else if(OperationVec[i]=='\*')

mult++;

else if(OperationVec[i]=='/')

div++;

}

elementsInside= plus+min+mult+div;

if(mult !=0 )

for(int **j**=OpStart;j>=OpEnd;j--)

if(OperationVec[OpEnd+j]=='\*')

{

j-=OpPlus;

ValuesVec[OpEnd+j+1]=ValuesVec[OpEnd+j] \* ValuesVec[OpEnd+j+1];

ValuesVec.erase(ValuesVec.begin() + j+OpEnd);

j+=OpPlus;

OperationVec.erase(OperationVec.begin()+j+OpEnd);

}

if(div !=0 )

for(int **j**=OpStart;j>=OpEnd;j--)

if(OperationVec[OpEnd+j]=='/')

{

if(ValuesVec[OpEnd+j+1]==0)

{

QMessageBox::information(this,"Error","Division by 0 is not possible");

Calculator::on\_pushButton\_reset\_clicked();

error=true;

}

else

{

j-=OpPlus;

ValuesVec[OpEnd+j+1]=ValuesVec[OpEnd+j] / ValuesVec[OpEnd+j+1];

ValuesVec.erase(ValuesVec.begin() + j+OpEnd);

j+=OpPlus;

OperationVec.erase(OperationVec.begin()+j+OpEnd);

}

}

if(plus !=0 )

{

for(int **j**=OpStart;j>=OpEnd;j--)

{

if(OperationVec[OpEnd+j] != '+') continue;

j -= OpPlus;

ValuesVec[OpEnd+j+1]=ValuesVec[OpEnd+j] + ValuesVec[OpEnd+j+1];

ValuesVec.erase(ValuesVec.begin() + j+OpEnd);

j+=OpPlus;

OperationVec.erase(OperationVec.begin()+j+OpEnd);

}

}

if(min !=0 )

for(int **j**=OpStart;j>=OpEnd;j--)

if(OperationVec[OpEnd+j]=='-')

{

j-=OpPlus;

ValuesVec[OpEnd+j+1]=ValuesVec[OpEnd+j] - ValuesVec[OpEnd+j+1];

ValuesVec.erase(ValuesVec.begin() + j+OpEnd);

j+=OpPlus;

OperationVec.erase(OperationVec.begin()+j+OpEnd);

}

}

void Calculator::**equ\_solution**()

{

double **temp**;

if(indicator !=0)

{

QMessageBox::information(this,"Error","One of the brackets is missing");

Calculator::on\_pushButton\_reset\_clicked();

error=true;

}

if(NumberOfBrace != 0)

{

int **bracketshead**=0, **bracketstail**=0;

while(NumberOfBrace!=0)

{

int **passed\_started**=0;

for(int **i**=0;i<OperationVec.size();i++)

if(OperationVec[i]==')')

{

bracketstail=i;

break;

}

for(int **i**=0;i<OperationVec.size();i++)

if(OperationVec[i]=='(' && i<bracketstail)

{

bracketshead=i;

passed\_started++;

}

if(OperationVec[0] != '(' && bracketshead==0)

{

QMessageBox::information(this,"Error","Incorrect brackets placement");

Calculator::on\_pushButton\_reset\_clicked();

error=true;

NumberOfBrace=0;

}

else

{

OpStart = bracketstail;

OpEnd = bracketshead;

OpPlus = passed\_started;

Calculator::equ\_solution\_res();

NumberOfBrace-=2;

OperationVec.erase(OperationVec.begin() + bracketstail-elementsInside);

OperationVec.erase(OperationVec.begin() + bracketshead);

bracketstail=0;

}

}

OpPlus=0;

}

if(OperationVec.size()!=0)

{

OpStart = OperationVec.size()-1;

OpEnd = 0;

Calculator::equ\_solution\_res();

}

}

void Calculator::**on\_actionnumeral\_system\_2\_triggered**()

{

this->close();

numeralsys->setWindowTitle("Calculator: NumeralSystem");

numeralsys->show();

}