Лабораторная работа №6.

Использование механизма ветвления и слияния системы управления версиями.

1. В репозитории приложения командной строки создать новую ветку разработки с названием exceptions.

1.1. Открыть проект приложения командной строки в QtCreator и сделать его активным.

1.2. Открыть панель управления ветками. Для этого выполнить команду Инструменты/Git/Локальное хранилище/Ветки.

1.3. В открывшемся окне создать новую ветку, нажав на кнопку Добавить и указав имя ветки exceptions. Убедиться, что ветка exceptions является текущей.

2. Заменить алгоритм обработки ошибок на другой, использующий исключения языка C++.

2.1. Подключить заголовок модуля стандартной библиотеки stdexcept:

```
#include <stdexcept>
```

2.2. Заключить весь код внутри функции main в блок try, разместив в блоке catch вывод сообщения о причине возникновения исключения:

```
try {
     // здесь поместить код программы
} catch (std::exception const & e) {
     std::cerr << e.what() << std::endl;
     return 1;
}
```

2.3. Заменить вывод сообщения об ошибке внутри операторов проверки условий на создание и генерацию стандартного исключения подходящего типа, с информативным сообщением о причине, например:

```
if(argc < 3) {
     throw std::length_error("Missing parameters");
}

if(astr.fail() || !astr.eof()) {
     throw std::invalid_argument("A must be an integer");
}

if(a <= 0) {
     throw std::out_of_range("A must be positive");
}
```

2.4. Проверить работу приложения в условиях возникновения исключительных ситуаций.

2.5. Зафиксировать изменения в репозитории.

3. Переключиться на главную ветку репозитория.

3.1. Открыть диалоговое окно управления ветками.

3.2. В локальном меню ветки master выбрать пункт "Сменить ветку" и подтвердить необходимость обновить открытые файлы в появившемся диалоговом окне.

3.3. Убедиться, что ветка master является текущей и содержимое исходных файлов проекта изменилось на версию до ветвления.

3.4. Переключиться обратно на ветку exceptions и убедиться, что сделанные в ней изменения также сохранились в репозитории.

3.5. Повторно переключиться на ветку master.

4. Добавить в проект перечисление для кодов ошибок, возвращаемых приложением.

4.1. Добавить новый перечислимый тип, содержащий константы для всех возможных значений кода ошибки с говорящими именами, например:

```
enum error_code {
     no_error = 0,
     not_enough_arguments = 1,
     invalid_type = 2,
     invalid_value = 3,
     invalid_range = 4
};
```

4.2. Заменить числовые литералы кодов ошибок в командах return на имена соответствующих констант.

4.3. Проверить работу приложения и сохранить изменения в репозитории.

4.4. Отобразить структуру репозитория в виде дерева. Для этого открыть окно интерпретатора командной строки в каталоге проекта и выполнить команду

```
git --no-pager log --graph --all
```

4.5. Убедиться, что структура репозитория имеет вид дерева с двумя ветками, в которых находятся сделанные коммиты.

5. Настроить инструмент слияния в системе управления версиями.

5.1. Убедиться, что утилита для слияния доступна из командной строки, выполнив команду

```
meld
```

в результате выполнения которой должно открыться диалоговое окно утилиты.

5.2. Указать meld в качестве инструмента для слияния в параметрах репозитория, выполнив команды

```
git config merge.tool meld
git config mergetool.meld.cmd "meld $LOCAL $BASE $REMOTE --output $MERGED"
```

5.3. Проконтролировать значения параметров конфигурации командой

```
git config --list
```

6. Выполнить слияние ветки exceptions с веткой master.

6.1. Убедиться, что в обеих ветках нет незафиксированных изменений, используя команду Инструменты/Git/Локальное хранилище/Состояние. Для переключения веток использовать окно управления ветками. Если есть незафиксированные изменения, их необходимо зафиксировать.

6.2. Убедиться, что ветка master является текущей. Название текущей ветки должно отображаться рядом с именем проекта в инспекторе "Проекты".

6.3. Выполнить процедуру слияния. Для этого в окне управления ветками в локальном меню ветки exceptions выбрать пункт "Объединить" и в открывшемся диалоговом окне нажать "Начать объединение".

6.4. В интерфейсе утилиты слияния устранить конфликты путем выбора нужного варианта либо непосредственного редактирования кода, после чего сохранить оба файла и закрыть окно.

6.5. Подтвердить успешное слияние, необходимость фиксации изменений и зафиксировать результаты слияния.

6.6. Отобразить структуру репозитория аналогично 4.4 и убедиться в том, что созданная ранее ветка exceptions соединилась с веткой master.

6.7. Добавить блоки обработки исключений для каждого типа исключений, используемых в блоке `try`, указав в операторе `return` соответствующий код ошибки, например

```
} catch (length_error const & e) {
    cerr << e.what() << endl;
    return not_enough_arguments;
} catch (invalid_argument const & e) {
    cerr << e.what() << endl;
    return invalid_type;
}
```

6.8. Проверить правильность работы программы и зафиксировать изменения в репозитории.

7. Перенести обработку ограничений на числовые значения границ (a > 0, b > 0, a < b) в библиотеку.

7.1. Вырезать соответствующие фрагменты из файла исходного кода приложения и вставить в файл реализации библиотеки.

7.2. Проверить работу приложения в условиях возникновения исключительных ситуаций и сохранить изменения в репозитории.

8. Самостоятельно аналогичным образом изменить графическое приложение.