Проект 7: Обработка на изходен код

Beautifier

- Тема:
    - Обработка на изходен код на C/C++

- Архитектура:
    - Начин на използване:
        - Под Windows от конзолата с текуща директория тази на програмата
            - Beautifier.exe <command> <command> <filename> …
        - Под Linux от конзолата с текуща директория тази на програмата
            - ./Beautifier <command> <command> <filename> …
    - Редът на командите и имената на файловете няма значение, както и броя им. След изпълнение на всички команди се записва нов файл със същото име завършващ с .old който съдържа стария неформатиран код, а във файла със същото име се записва форматирания код. Тази функционалност е различна само когато е използвана командата –html - при този случай не се записват промените направени във същия файл и не се записва .old файл, единствено се записва файл със същото име и .html разшерение форматиран по подходящ начин описан по-долу.
    - Команди
        - --help
            - Извежда информация за начина на работа с програмата
        - --html
            - Прилага се след всички други файлове.
            - Не приема аргументи.
            - Създава нов файл със същото име, но с разшерение .html който съдържа кода форматиран във вид който може да се отвори като html страница (индентиран, оцветен, и тн.).
            - Използва опциите зададени в cpp.ini файла за да създаде html файла с желаните оформления зададени в cpp.ini.
        - --comments
            - Не приема аргументи.
            - Премахва всички коментари от вида /* коментар */ и //коментар
        - --indentation
            - Приема един от следните аргументи tabs, spaces

        - --indentation=tabs – променя всичките индентации във файла на табулации, т.е ще замени всички 4 паузи с табулация, а табулациите ще си останат табулации.
        - --indentation=spaces – променя всичките индентации във файла на 4 паузи, т.е ще замени всички табулации с 4 паузи.
    - --format
        - Не приема аргументи.
        - Форматира файла по подходящ начин – слага индентации където е нужно, маха ги където не е нужно, всички нови сегменти от кода са на нов ред, изчиства излишните нови редове.
    - --newlines
        - Приема един от следните аргументи CRLF, LF, CR
        - --newlines=CRLF – променя всеки край на нов ред да бъде \r\n по стандартите на Windows за край на ред във файл.
        - --newlines=CR – променя всеки край на нов ред да бъде \r по стандартите на Mac за край на ред във файл.
        - --newlines=LF – променя всеки край на нов ред да бъде \n по стандартите на UNIX и Linux за край на ред във файл.

- Начин на работа на програмата (за някои функции и класове)
    - String – много от медотите на класа String са излишни, като нпример всички методи които приемат и/или връщат Pair или Vector от Pair тъй като те бяха използвани при по-ранна версия на програмата, но не работеше ефективно и бяха сменени с друг подход.
    - Beautify и Html от класа CodeFormatter работят по следния начин:
      Итерира се по целия код и се дигат флагове когато итераторът е в кавички, коментар и тн. В Html() това се ползва за да се оцветят по съответен начин, а в Beautify() се използва за да не се прилага индентация вътре във коментарите или кавичките и тн.
      Html() итерира веднъж целия код и слага където е необходими html тагове.
      Beautify() използва стек за да държи всички вложени блокове код (if, else, for, etc.)
      Подходът ми към тези функции беше избран защото при предишната реализация (когато не бяха по 200-300 реда код) сложността на алгоритъма беше експоненциална и не ефективна. За това във финалния си вариант тези функции са много по-дълги от добрите практики, но алгоритъма е много по-ефикасен.

- Основни класове:

  По подробна информация за класовете може да се намери в UML-диаграмата. Не описаните функции и полета могат да се разглеждат като black-box тъй като са твърде много и са повече помощни функции и временни променливи спомагащи работата.

  - Pair – Наподобява стандартната реализация
  - Trio – Наподобява стандартната реализация
  - Stack – Наподобява стандартната реализация
  - String – Наподобява стандартната реализация
  - Vector – Наподобява стандартната реализация
  - CodeFormatter
    - Член данни
      - Vector<String> extensions – съдържа всички разрешени разширения на файлове за обработа от избрания конфиг. файл
      - Vector<String> brackets – съдържа всички скоби разрешени от езика от избрания конфиг. файл
      - Vector<String> operators – съдържа всички оператори от езика от избрания конфиг. файл
      - Vector<String> keywords – съдържа всички запазени думи от езика от избрания конфиг файл
      - Vector<String> keywords_with_indent – съдържа всички запазени думи от езика след които следва индентиран блок (if, else, etc.)
      - Vector<String> keywords_with_brackets – съдържа всички запазени думи от езика след които трябва да следва условие в скоби () (if, for, etc.)
      - String codeFile – името на файла с оригиналния код
      - String configFile – името на конфиг. файла
      - String codeOriginal – оригиналния код
      - String codeFormatted – форматирания код
      - String config – конфигурацията като стринг
      - String html – html кодът се записва в тази променлива ако потребителя е използвал –html
    - Методи
      - String CodeFile() – връжа името на файла с код, върху който се работи
      - String ConfigFile() – връща името на файа с конфигурацията, която се използва

- String Code() – връща форматирания код
- String Config() – връща конфигурацията като стринг
- Vector<String> Extensions() – връща всички разрешени разширения от конфигурацията
- void Code(const String& Cod) – променя съдържанието на форматирания код (оригиналния код не се променя)
- void Config(const String& Conf) – задава нова конфигурация от стринг
- void CodeFile(const String& Name) – задава ново име на файла с код
- void ConfigFile(const String& Name) – задава ново име на конфиг. файла
- void LoadCode() – прочита целия код от файла codeFile в codeOriginal
- void LoadConfig() – прочита цялата конфигурация от файла configFile в config
- void Save() – запазва форматирания код във codeFile
- void SaveOriginal(String) запазва оригиналния код в посочен от потребителя файл
- void SaveAs(String) – запазва форматирания код в посочен от потребителя файл
- void SaveHtmlAs(String Name) – запазва html стринга в посочен от потребителя файл (може да е празен стринг ако не е извинака командата –html)
- void RemoveComments() – премахва всички коментари
- void ChangeNewLines(const String Format) – променя формата на всички нови редове
- void ChangeIndentation(const String Format) – сменя индентацията във файла от паузи на табулации и обратно
- void Beautify() – индентира кода
- void Html() – генерира html файл с кода оцветен, индентиран и тн. така че да може да се разглежда във всеки браузър

- File
  - Член данни
    - String name – съдържа името на файла върху който се работи
  - Методи
    - String Name() – връща името на файла
    - void Name(String) – задава ново име на файла
    - bool ReadAll(String&) – чете цялата информация на файл и я записва в подадения стринг

        - bool WriteAll(String&) – записва цялата информация от подадения стринг във файла
        - bool AppendAll(String&) – записва цялата информация от подадения стринг във файла без да трие каквото е имало преди това

- Използван чужд код:
    - Библиотеките на C++
        - `<fstream>`
        - `<iostream>`
        - `<iomanip>`
    - Библиотеките за тестване
        - `<cstdlib>`
        - `<crtdbg.h>`