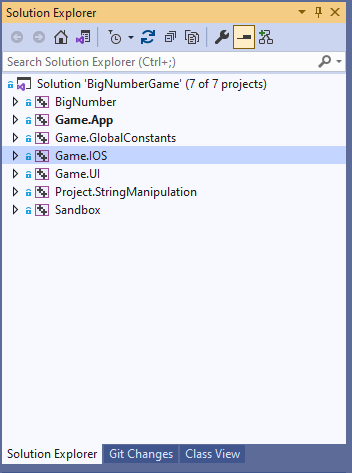
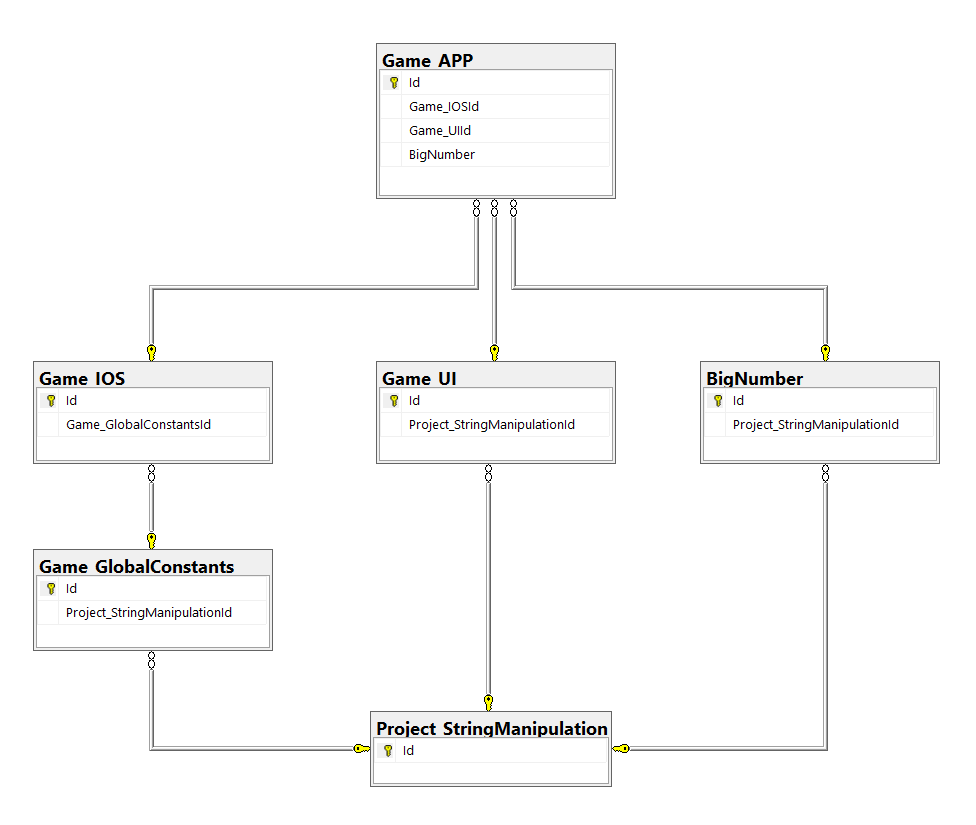
Документация на *BigNumberGame* проект №2

1. **Описание и идея на проекта**
   1. Идеята на проекта е реализацията на проста игра, в която играчът трябва да изчислява изрази с големи числа. Тези големи числа идват под формата на BigNumber класовете от проект №1 (линк към неговата документация в GitHub има по-долу в графа Github).
   2. В процеса на работа се оказа наложимо да си направя и собствена мини-библиотека за работа със *cstrings* с цел по-лесно боравене с тях и улесняване на работата си по изграждане на функционалността на *BigNumber* и т.н. Тази библиотека е изработена лично от мен на ръка, без копиране на чуждестранен код, така че и тя също влиза в проекта, макар и като по-скоро страничен ефект от създаването на останалите части на проекта, отколкото като конкретно поставена задача.
   3. По-късно обаче осъзнах, че работата със cstrings е досадна дори и тази готова библиотека. Затова създадох свой собствен клас MString(mstring) и създадох нова библиотека, която да работи със този клас MStringManip.
   4. Съответно, на почти всяко едно място в програма, работата със cstring беше заменена със този клас.
   5. В проекта влизат и 5 различни тестови класове, които имат за цел да тестват всеки един от съответните класове за тяхната коректна работа. 3 от тях идват от проект №1 и гарантират коректна работа на BigNumber. 4-ти и 5-ти тестови класове са MStringTests и MStringManipTests, които гарантират коректната работа съответно на MString и MstringManip.
   6. Старата библиотека за работа със cstrings също е включена във сорс кода. Не ми даваше сърце да я изтрия. Но реално не се използва в рамките на този проект
2. **Основни дефиниции, концепции и алгоритми, които ще бъдат използвани**
   1. Като концепции използвам повечето неща, взети на лекции.
   2. Използва се работа с множество обекти и принципът за ООП.
   3. Използва се основната концепция за *Big 4* при работа с динамична памет. Там, където е нужно, е имплементирана и тествана.
   4. В тази връзка естествено се ползват и познанията за работа с динамична памет и със референции и препратки. Където е необходимо, динамичната памет се освобождава, за да не възникват *memory-leak* проблеми.
   5. Приложена е концепцията за преизползваемост на кода. Тоест, в повечето случаи, където има повтарящ се код, този код се изнася като отделна функция или в един конкретен случай като отделен клас с цел леснота при поддръжка и коригиране на този код в случай на установен проблем в него.
   6. Това се отнася до имплементацията на същинските класове като *BigNumber*, *BigNumberExpression*, *GameController, FileSystem* и т.н. Специално изключение правят тестовите класове, където съм пресметнал за по-добре да не изнасям в отделни функции общата функционалност между отделни *test cases* за една и съща функционалност.
3. **Структура на обектите в проекта**
   1. Проектът е написан на Visual Studio. Отдолу са показани отделните компоненти на *BigNumberGame* проектът.

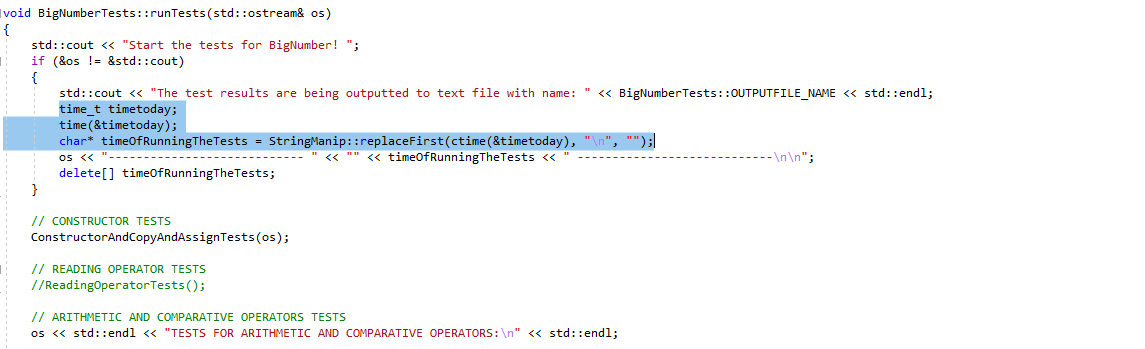


Снимка 1

* 1. За реализацията на този проект съм създал 6+1 проекта във Visual Studio.  
     1. *Project.StringManipulation* проектът е библиотека, съдържаща в себе си собствена реализация на string клас, както и помощни функции за работа със тези string-ове. Съдържа още и 2 тестови класа, чрез които MString и MStringManip се тестват за коректна работа двата.
     2. *BigNumber* проектът съдържа в себе си всичката функционалност за *BigNumber* и *BigNumberExpression* и *BigNumberEquation*, както и за тяхното тестване.
     3. Game.GlobalConstants проектът съдържа много на брой константни променливи, които определят важни за програмата неща, най-вече свързани с UI частта. Например, в нея чрез константи са зададени различните команди в програмата и техните параметри.
     4. Game.UI проектът се грижи за директната работа с UI на програмата. Тоест, в нея има различни константи и функции, които директно принтират на конзолата по различни начини някакъв текст.
     5. Game.IOS проектът се грижи за директната работа с базата данни на програмата, тоест за директната работа със файлове. В нея има най-различни CRUD операции.
     6. Game.App проектът се грижи за програмния цикъл и състоянието на играта. Тоест, в него се определя на кой текущ екран е играчът и какви команди има на разположение. Спрямо тези команди проектът прави извиквания до базата данни чрез Game.IOS и чрез тези данни извежда нещо на екрана чрез Game.UI или смяна състоянието на играта като изпраща играчът на друг екран.
     7. Sandbox проектът не влиза в реалната функционалност на играта. Идеята на този проектът е в него да мога да експериментирам по какъвто начин си искам с останалите компоненти на програмата, ако е наложително.
  2. *Project.StringManipulation* проектът съдържа в себе си:  
     1. Класът MString, който представлява собствена имплементация на вече съществуващия string клас в c++. MString класът в програмата най-често се използва като mstring за по-лесно писане(чрез typedef MString mstring).
     2. Класът M*StringManip*, който съдържа в себе си помощните функции за работа със mstring.
     3. Класът MStringTests, който съдържа в себе си тестова за работата на mstring.
     4. Класът *StringManipTests*, който съдържа в себе си тестове за повечето функции на *StringManip.*
  3. *BigNumber* проектът съдържа в себе си:  
     1. *BigNumber* класът, който в себе си съдържа цялата функционалност за работа с големи числа.
     2. *BigNumberDivisionResult* класът, който съдържа информацията за частното и остатъка от някакво деление на големи числа.
     3. *BigNumberTests* класът, който съдържа в себе си всички тестове за *BigNumber.*
     4. *BigNumberExpression* класът, който съдържа функционалността за изрази, съставени от големи числа, в себе си.
     5. *BigNumberExpressionTests* класът, който в себе си съдържа тестовете за *BigNumberExpression.*
     6. *BigNumberEquation* класът, който в себе си съдържа функционалността за уравнения, съставени от големи числа.
     7. *BigNumberEquationTests* класът, който в себе си съдържа тестовете за *BigNumberEquation.*
     8. *BigNumberExpressionCommon* класът, който съдържа в себе си виртуални функции, необходими за генерирането на случаен принцип на компоненти от изразите и уравненията. *BigNumberExpression* и *BigNumberEquation* класовете го наследяват.
  4. Game.GlobalConstants проектът съдържа в себе си:  
     1. Класът GlobalConstants, който съдържа в себе си важни за програмата константни променливи.
  5. Game.UI проектът съдържа в себе си:  
     1. Класът GameUI, който съдържа в себе си различни константни променливи и статични функции, които позволяват принтиране на информация на конзолата по различни начини, например под формата на екран или на таблица.
     2. Енъмът TextAlign, който в някои случаи определя приравняването на текста в един ред на конзолата.
  6. Game.IOS проектът съдържа в себе си:  
     1. Класът User, който съдържа в себе си данните от неговите колони в базата данни като полета.   
        Съдържа още и константни променливи с важна информация за потребителите, като броя на колоните му, какви роли може да има, позволените дължини на потребителските имена и паролите и exception съобщенията при нарушаване на някои от тези ограничения.  
        Освен това класът съдържа функции, които за определени полета от класа да връщат тяхната стойност, представена като string във формат, подходящ за записване в базата данни.
     2. Класът DeletionMessage, който съдържа в себе си данните от неговите колони в базата данни като полета.   
        Освен това класът съдържа функции, които за определени полета от класа да връщат тяхната стойност, представена като string във формат, подходящ за записване в базата данни.
     3. Класът FileSystem, който съдържа в себе си различни функции за работа с файловата система, които са общи за всички останали класове, които трябва да работят с CRUD операции върху базата данни. Той не е родител на тези класове.  
        Също съдържа константни променливи с важна информация за базата данни. Например съдържа разделителите за колона и за запис в базата данни, както и текстът за празна(null) стойност на колона в базата данни.
     4. Класът UserTable, който съдържа в себе си функции за валидация на потребителите и за CRUD операции върху таблицата от базата данни, съдържаща потребителите.
     5. Класът DeletionMessageTable, който съдържа в себе си функции за CRUD операции върху таблицата от базата данни, съдържаща съобщенията за изтриване на потребителите.
     6. Класът Seeder, който съдържа в себе си функции за сийдване на базата с предварително определени данни, включително и тези за потребителите админи, които могат да се създават единствено по този начин.
  7. Game.App проектът съдържа:  
     1. Класът Controller, който съдържа в себе си полета и функции, които са общи за 2 или повече други контролери. Например съдържа информация за текущия потребител, ако не е анонимен(тоест не е влязъл още в профила си). Също съдържа и инициализиращата функция initialize().   
        Този клас не е родител на другите контролери.
     2. Класът UnauthenticatedController, който съдържа в себе си функции за различните екрани на анонимни потребители, както и функции, които за всеки екран принтират съответният текст. В тези функции се работи с класовете от Game.IOS и Game.UI, за да се извлекат данни, да се валидират данни и команди и да се извеждат на екрана съответните неща.
     3. Класът AuthenticatedController, който съдържа в себе си функции за различните екрани на логнати потребители, както и функции, които за всеки екран принтират съответният текст. В тези функции се работи с класовете от Game.IOS и Game.UI, за да се извлекат данни, да се валидират данни и команди и да се извеждат на екрана съответните неща.
  8. Диаграма на зависимостите на различните проекти в програмата:

Диаграма 1

* 1. Зависимостите с думи:   
     1. Project.StringManipulation не зависи от никой друг проект.
     2. Game.GlobalConstants зависи от Project.StringManipulation.
     3. Game.UI зависи от Project.StringManipulation
     4. BigNumber зависи от Project.StringManipulation
     5. Game.IOS зависи от Project.StringManipulation и Game.GlobalConstants
     6. Game.App зависи от всички други проекти

1. **Реализация и тестове**
   1. Погледнете в GitHub страницата с документацията на проекта. И по точно в README файлът.
   2. Погледнете в *Doxygen* генерираната документация.
   3. В сорс кодът на проекта.
   4. За библиотеката за стрингове и за BigNumber имам директно реализирани тестове, които обаче не са unit тестове, използващи външна библиотека.
   5. Останалата част на проекта е тествана на ръка. Когато променя нещо, обмислям всички крайни случаи и тествам първо за тях, след което тествам и за обща коректност на това, което съм променил.
2. **GitHub**
   1. <https://github.com/rosenkolev1/University-of-Sofia-cplusplus-exercises>   
      Това е цялото ми *repo*, което съдържа всички неща, свързани с програмирането в университета, включително и BigNumberGame проекта.
   2. <https://github.com/rosenkolev1/Project-1-BigNumberDocumentation>  
      Това е репото с документацията за проект №1 на BigNumber. Документацията към днешният момент е леко остаряла поради рефакторирането на библиотеката за работа със стрингове и на BigNumber, който вече използва тези нови стрингове.
   3. <https://github.com/rosenkolev1/University-of-Sofia-cplusplus-exercises/tree/main/Computer%20Science%20Year-1%20Term-2/BigNumberGame>  
      Това е директорията на *solution-на*, в който се намират цялата програма. Същият *solution* е показан и на „снимка 1“.  
      В тази папка се намират и оригиналните имплементации на BigNumber и на StringManip, които работят със cstrings вместо със mstring класа.
   4. TODO: ADD THE DIRECTORY TO THE GITHUB DOC FOR PROJECT 2
3. **Източници**
   1. Типичните сайтове, които излизат в търсачката на гугъл, когато попитаме как се прави нещо на c++: *geeksforgeeks.com*, *cppreference.com*, *stackoverflow.com*, в някои случай, понеже пиша на visual studio, и майкрософтската документация, и т.н.
   2. Чужд код обаче съм копирал точно един път. Ето го и него:
   3. Кодът има за цел да изпише текущото време на стартиране на тестовете. Това не знаех как става в c++, за това погледнах в интернет и частта с *time\_t* от *stackoverflow*.
   4. Дори и за използването на *srand* и *rand* не ми трябваше да копирам от интернет, понеже вече имах от преди това програма, в която се използваше. Мисля че беше от второто домашно от първия семестър.

*Изработил: Росен Андреев Колев, 4-та група, 0MI0800065*