Контекстно-свободни граматики

1. Увод

1.1.Описание и идея на проекта

Проектът реализира основни алгоритми и действия върху контекстно-свободни граматики, използващи главни латински букви за променливи (нетерминали) и малки латински букви за терминали. Информацията за наличните граматики се поддържа във файл

1.2.Цели и задачи

1.2.1. Представяне и обработване на информацията за правило

1.2.2. Представяне и обработване на информацията за граматика и изпълняване на множество операции с нея

1.2.3. Прочитане на данни от файлове и записване на направените промени във файл съответно за граматиките

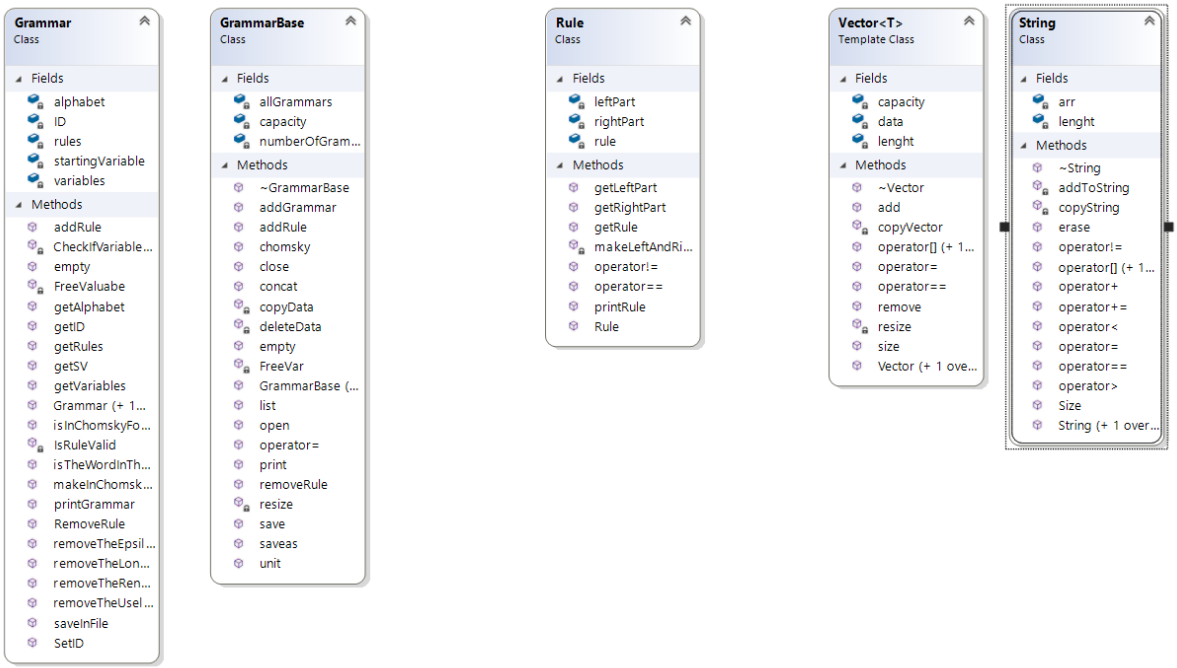
1.3. Структура на документацията

1.3.1. Увод

1.3.2. Проектиране

1.3.3. Реализация

2. Проектиране



За реализацията са използвани следните класове

- class Vector<T> - шаблонен вектор
  Има следните член-данни:
  - T* data – динамичен масив от тип T

- int capacity – капацитет на масива
- int lenght – дължина на масива

Има следните private методи:

- void resize() – удвоява капацитета на масива
- void copyVector(const Vector<T>&) – копира информацията за вектор

Има следните методи:

- Vector(int = 1) - контруктор
- Vector(const Vector<T>&) – копи-конструктор
- Vector<T>& operator=(const Vector<T>&) – оператор =
- ~Vector() - деструктор
- int size() const – връща големината на масива
- void add(const T&) – добавя елемент в масива
- void remove(const T&) – изтрива елемент от масива
- bool operator==(const Vector<T>&) – оператор ==
- T& operator[](const int) - оператор[]
- const T& operator[](const int)const – оператор[]

- class String
  Има следните член-данни:
  - char* arr – динамичен масив от символи
  - int lenght – дължина на масива

  Има следните private методи:

  - void copyString(const String&) – копира информацията за стринга
  - void addToString(const char) – добавя елемент към стринга

  Има следните методи:

  - String(const char* = "") - конструктор
  - String(const String&) – копи-конструктор
  - String& operator=(const String&) – оператор =
  - ~String() - деструктор
  - int Size()const – връща дължината на масива
  - bool operator==(const String&) const – оператор ==
  - char& operator[](const int) – оператор []
  - const char& operator[](const int)const -оператор[]
  - String operator+(const String&) – оператор +
  - String& operator+=(const String&) – оператор +=
  - bool operator>(const String&) const – оператор >
  - bool operator<(const String&) const – оператор <
  - friend std::istream& operator>>(std::istream&, String&) – оператор за въвеждане >>
  - friend std::ostream& operator<<(std::ostream&, const String&) – оператор за извеждане <<

- friend void getline(std::istream&, String&) – оператор за въвеждане getline с финален символ нов ред по подразбиране
- friend void getline(std::istream&, String&, char) – оператор за въвеждане getline

- class Rule – съхранява информацията за правило
  Има следните член-данни:
  - String rule – низ с цялото правило
  - String leftPart – частта от правилото преди ->
  - String rightPart – частта от правилото след ->

  Има следния private метод:

  - void makeLeftAndRightPart() – образува leftPart и rightPart

  Има следните методи:

  - Rule(const String = "") – конструктор (в него се правилото се разделя на две части)
  - String getLeftPart()const – връща лявата страна на правилото
  - String getRightPart()const – връща дясната страна на правилото
  - String getRule()const – връща цялото правило
  - void printRule()const – извежда правилото
  - bool operator==(const Rule&)const – оператор ==
  - bool operator!=(const Rule& other)const – оператор !=

❖ class Grammar – съхранява информация за граматика
  Има следните член-данни:
  - Vector<char> variables – променливите в граматиката
  - Vector<char> alphabet – азбуката в граматиката
  - Vector<Rule> rules – правилата в граматиката
  - char startingVariable – началната променлива
  - int ID – уникален номер

  Има следните private методи:
  - bool CheckIfVariableIsInArray(String, Vector<String>) – проверява дали някоя част от правило съдържа дадена променлива
  - bool IsRuleValid(Vector<char>, Rule) – проверява дали променливите в правилото са тези и на граматиката
  - char FreeValuabe(Vector<char>) – връща първата неизползвана главна латинска буква за променлива

  Има следните методи:
  - Grammar() – конструктор по подразбиране
  - Grammar(Vector<char>, Vector<char>, Vector<Rule>, char) – конструктор с параметри
  - Grammar removeTheUselessVariables() – премахва безполезните променливи
  - Grammar removeTheLongRules() – премахва дългите правила
  - Grammar removeTheRenamingRules() – премахва преименуващите правила(нереализиран метод)

- Grammar removeTheEpsilonRules() – премахва епсилон правилата (нереализиран метод)
- Grammar makeInChomskyForm() – прави граматиката в нормална форма на Чомски (нереализиран метод)
- void printGrammar() const – извежда информация за граматиката
- void addRule(const Rule&) – добавя правило към граматиката
- void RemoveRule(const Rule&) – изтрива правило от граматиката
- bool isInChomskyForm() – проверява дали граматиката е във формата на Чомски
- bool isTheWordInTheLanguage(const String&) – проверява дали дума се съдържа в езика на граматиката(нереализиран метод)
- bool empty() – проверява дали езикът на граматиката е празен
- friend std::istream& operator>>(std::istream&, Grammar&) – оператор за въвеждане >>
- void SetID(const int index) – определя уникалния номер
- int getID()const – връща уникалния номер
- Vector<Rule> getRules()const – връща правилата на граматиката
- Vector<char> getVariables() – връща променливите на граматиката
- Vector<char> getAlphabet()const – връща азбуката на граматиката
- char getSV()const – връща началната променлива на граматиката
- friend void saveFromFile(std::istream&, Grammar&) – чете информацията от файл
- void saveInFile(std::ostream&) – записва информацията от файл

❖ class GrammarBase – функция на вектор от граматики, но със разширена функционалност

Има следните член-данни:

- Grammar* allGrammars – масив от граматики
- int capacity – капацитет на масива
- int numberOfGrammars – граматики в масива

Има следните private методи:

- void copyData(const GrammarBase&) – копира информация за наличните граматики
- void deleteData() – изтрива динамично заделената памет
- void resize() – удвоява капацитета на масива
- char FreeVar(Vector<char>, Vector<char>) – връща променлива, която не се среща в двата подадени масива

Има следните методи:

- GrammarBase(int = 1) - конструктор
- GrammarBase(const GrammarBase&) – копи-конструктор
- GrammarBase& operator=(const GrammarBase&) – оператор =
- ~GrammarBase() - деструктор
- void list()const – извежда всички уникални номера на граматиките

- void print(int) const – извежда информация за граматиката с подадения уникален номер
- void addRule(int, const Rule&) – добавя правило към дадена граматика
- void removeRule(int, int) – премахва правило с подаден номер от дадена граматика
- bool chomsky(int)const – проверява дали дадена граматика е във формата на Чомски
- bool empty(int)const – проверява дали езикът на дадена граматика е празен
- void addGrammar(const Grammar&) – добавя граматика към масива(тук се генерира уникалния номер на граматика)
- Grammar concat(int, int) – конкатенира две граматики, създава нова и я записва в масива
- Grammar unit(int, int) - обединява две граматики, създава нова и я записва в масива
- void open(const char*) – отваря файл и прочита информацията за граматики от него
- void save(const char*) – запазва направените промени в последно отворилия се файл
- void saveas(const char*) – записва информацията в нов файл
- void close() - изтрива заредената досега информация

3. Реализация

- Информацията във файл се записва по следния начин:

БройНаПроменливи БройНаБуквитеВАзбуката БройНаПравилата
Пр
оменливи
Азбука
Правила
Начална променлива

Примерен файл – grammars.txt
- Вместо символът за епсилон в правилото се записва eps
- Едно правило се записва от вида: A->… . Програмата не поддържа правила от вида A->…|…|….
- Реализациите на някои функции от условието(chomskufy<id>, cyk<id>, iter<id>) липсват