|  |  |
| --- | --- |
| Программа для преобразования неравенства к операции «меньше» | |
| Внутренняя спецификация | |
| Студент | Еремеев М.А. |
| Преподаватель | доц. Сычев О.А. |
| Сдано |  |

1. Назначение

Программа предназначена для преобразования неравенства к операции «меньше».

2. Описание структур данных

Перечисление ExprNodeType содержит значения: Plus, Minus, Division, Multiplication, DivRemainder, UnaryMinus, Not, GreaterEqual, LessEqual, Greater, Less, Equal, Operand.

Перечисление ErrorType содержит значения: inFileNotExist, emptyFile, moreThenOneLineInFile, invalidRootOperator, unknownSymbol, missingOperand, redundantOperand.

Класс Error содержит поля:

ErrorType type – тип ошибки

std::string errorInputFilePath - поле для хранения пути, откуда не удалось

прочитать файл

string strWithError – строка, содержащая ошибку

int position – позиция ошибки

string generateErrorMessage() – метод, возвращающий сообщение с описанием ошибки

Класс ExprNode содержит поля:

ExprNodeType type – тип узла выражения

string value – значение узла

ExprNode\* firstOperand – первый операнд

ExprNode\* secondOperand – второй операнд

static const map <string, ExprNodeType> symbolToString – статический словарь для получения словесной записи оператора по его символу

static const map <ExprNodeType, string> stringToSymbol – статический словарь для получения символа оператора по его словесной записи

static const map <ExprNodeType, int> operandCount - статический словарь для получения количества операндов

static const set <string> operatorsSymbols – статический словарь с символами

void swapOperands() – метод для изменения мест операндов

void addUnaryOperatorBefore ( ExprNodeType operator) – метод для добавления унарной операции перед текущим узлом

ExprNode \* copyNode () – метод для копирования узла

string getRpmOfTree() – метод получения обратной польской записи узла

3. Описание алгоритмов функций

Главная функция программы:

int main(int argc, char\* argv[]);

Входные данные: argc - количество переданных аргументов командной

строки argv[] - аргументы командной строки: argv[1] - путь к

проектному файлу, argv[2] - путь к выходной директории

Выходные данные:

0 – функция завершилась успешно

Алгоритм функции:

Завершить программу, если получено меньше трех параметров

Считать из параметров командной строки путь к входному файлу

Считать из параметров командной строки путь для выходных файлов

Получить обратную польскую запись выражения (readFileToString)

Если удалось открыть входной файл и ошибок не было обнаружено

Получить дерево разбора выражения (stringToTree)

Если дерево разбора построено без ошибок

Проверить корневую операцию (checkRootOperator)

Создать выходной файл

Если удалось считать данные без ошибок и файл для выходных данных был создан

Преобразовать к операции меньше (transformInequalityToLessOperator)

Получить обратную польскую запись измененного дерева (getRpnOfTree)

Вернуть в выходной файл измененное неравенство в обратной польской записи

Очистить память, выделенную под дерево и строку с обратной польской записью

Если выходной файл был создан и есть ошибки

Для каждой ошибки

Записать в выходной файл информацию об ошибке (generateErrorMessage)

Иначе если выходной файл не был создан

Вывести в консоль ошибку о неудаче создания выходного файла

Завершить программу

Закрыть выходной файл

Завершить работу программы

Функция программы:

std::string \* readFileToString(const string filePath, std::set <Error> \* errors)

Входные данные: filePath– путь до файла, откуда будут прочитаны данные

Выходные данные: строка с содержимым файла; errors – указатель на контейнер с ошибками, при их наличии

Алгоритм функции:

Открыть файл

Если не удалось открыть файл

Добавить ошибку о неудаче открыть файл

Иначе

Считать непустые строки файла

Закрыть файл

Если в файле все строки пустые

Добавить ошибку о пустом входном файле

Если в файле больше одной непустой строки

Для каждой непустой строки

Добавить ошибку о том, что в файле больше одной непустой строки

Если нет ошибок

Считать содержимое первой непустой строки в поток

Вернуть содержимое потока

Вернуть nullptr

Функция программы:

ExprNode \* stringToTree(std::string rpnString, std::set <Error> \* errors)

Входные данные: rpnString – строка с выражением в обратной польской записи

Выходные данные: указатель на корневой узел дерева разбора выражения; errors – указатель на контейнер с ошибками, при их наличии

Алгоритм функции:

Создать стэк с содержимым класса ExprNode

Создать поток с выражением в обратной польской записи

Создать словарь для позиций токенов в исходной строке

Пока в переменную token получается извлечь часть потока до первого пробельного символа и нет ошибок

Создать узел newNode

Добавить в словарь позицию токена

Если token – операнд (isOperand)

Присвоить полю value newNode значение token

Присвоить полю type newNode значение Operand

Иначе если token – символ из словаря symbolToString

Присвоить значению type newNode словесную запись оператора из словаря symbolToString

Узнать необходимое количество операндов из словаря operandCount

Если в стеке нет нужного количества операндов

Добавить ошибку о недостатке операндов

Если нет ошибок

Если оператор бинарный

Присвоить второму операнду newNode значение верхнего узла из стека

Удалить из стека верхний узел

Присвоить первому операнду newNode значение верхнего узла из стека

Удалить из стека верхний узел

Иначе

Добавить ошибку о недопустимом символе

Если нет ошибок

Поместить newNode в стэк

Если в стэке больше одного элемента

Сохранить верхушку стека

Для всех неиспользованных операндов в стэке

Добавить ошибку о наличии лишнего операнда

Если нет ошибок

Вернуть верхушку стэка (stack.top())

Иначе

Освободить память, выделенную под стэк

Вернуть nullptr

Функция программы:

void transformInequalityToLessOperator( ExprNode \* node)

Входные данные: node – ссылка на корневой узел дерева разбора выражения

Выходные данные: node – ссылка на корневой узел измененного дерева разбора выражения, преобразованного к операции «меньше»;

Алгоритм функции:

Если корневая операция отличается от “Less”

Запомнить корневую операцию в originalOperator

Заменить корневую операцию на “Less”

Switch originalOperator:

Case “Greater”:

Поменять операнды местами (swapOperands)

Case “GreaterEqual”:

Добавить новую операцию “Not” перед корневой (addUnaryOperatorBefore)

Case “LessEqual”:

Поменять операнды местами (swapOperands)

Добавить операцию “Not” перед корневой (addUnaryOperatorBefore)

Функция программы:

void ExprNode :: swapOperands()

Входные данные:

Выходные данные:

Алгоритм функции:

Сохранить указатель на первый операнд в tmp

Поместить в поле для первого операнда указатель на второй операнд

Поместить в поле для второго операнда tmp

Функция программы:

void ExprNode :: addUnaryOperatorBefore ( ExprNodeType \_operator)

Входные данные: \_operator – тип операции, которая будет добавлена пере текущей

Выходные данные:

Алгоритм функции:

Создать указатель на новый узел newNode

Копировать текущий узел в newNode (copyNode)

Поместить в первый операнд текущего узла newNode

Обнулить значение второго операнда текущего узла

Изменить тип операции на operator

Функция программы:

ExprNode \* ExprNode :: copyNode ()

Входные данные:

Выходные данные: копия узла

Алгоритм функции

Создать указатель на новый узел newNode

Присвоить тип операции текущего узла в поле type узла newNode

Если текущий узел – операнд

Присвоить значение операнда в поле value узла

Иначе если текущий узел имеет первый операнд

Скопировать в первый операнд newNode первый операнд текущего узла

Присвоить в первый операнд newNode значение первого операнда текущего узла

Если текущий узел имеет второй операнд

Присвоить в второй операнд newNode значение второго операнда текущего узла

Вернуть newNode

Функция программы:

bool checkRootOperator( ExprNodeType \_operator)

Входные данные: operator – тип операции, которая будет проверена

Выходные данные: bool допустим ли корневой оператор

Алгоритм функции:

Если \_operator присутствует в контейнере допустимых корневых операторов {Less, Greater, GreaterEqual, LessEqual}

Вернуть true

Иначе

Добавить ошибку о недопустимом корневом операторе

Вернуть false

Функция программы:

bool isOperand( std::string str)

Входные данные: str – строка с операндом

Выходные данные: является ли строка операндом

Алгоритм функции:

Считать строку операндом

Если длина str больше 1 и первый символ str - «0»

Если второй символ str - «X» или «x»

Считать str не операндом, если хотя бы один символ недопустимы в шестнадцатеричной системе счисления

Иначе если второй символ str – число

Считать str не операндом, если хотя бы один символ – не цифра меньше 8

Иначе

Считать строку не операндом

Иначе если первый символ – цифра

Считать строку не операндом, если есть хотя бы один нецифровой символ

Если str начинается с буквы или «\_»

Считать str не операндом, если хотя бы один символ - не цифра и не буква

Вернуть, является ли str операндом

Рекурсивная функция программы:

string ExprNode :: getRpmOfTree()

Входные данные:

Выходные данные: обратная польская запись текущего узла

Алгоритм функции

Создать строку для хранения записи текущего узла

Если текущий узел – операнд

Записать в строку значение поля value

Иначе если текущий узел имеет первый операнд

Записать в строку обратную польскую запись первого операнда (getRpnOfTree)

Если текущий узел имеет второй операнд

Добавить к строке обратную польскую запись второго операнда (getRpnOfTree)

Добавить к строке математическую запись оператора из словаря stringToSymbol

Вернуть строку с обратной польской записью

Функция программы:

string generateErrorMessage()

Входные данные:

Выходные данные: строка с описанием ошибки

Алгоритм функции

Создать строку message для хранения текста ошибки

switch type:

case inFileNotExist:

Добавить в message сообщение о неудаче открытия файла по пути errorInputFilePath

case emptyFile:

Добавить в message сообщение о пустом входном файле

case moreThenOneLineInFile:

Добавить в message сообщение о лишней строке с содержимым strWithError

case invalidRootOperator:

Добавить в message сообщение о недопустимой корневой операции strWithError

case unknownSymbol:

Добавить в message сообщение о неизвестном символе strWithError

case missingOperand:

Добавить в message сообщение о лишнем операторе strWithError

case redundantOperand:

Добавить в message сообщение о лишнем операнде strWithError

Вернуть message

4. Диаграмма вызовов функций и UML-диаграмма классов приведена в приложении №1.

5. Диаграмма потоков данных приведена в приложении №2.

Приложение №1 - Диаграмма вызовов функций и UML-диаграмма классов

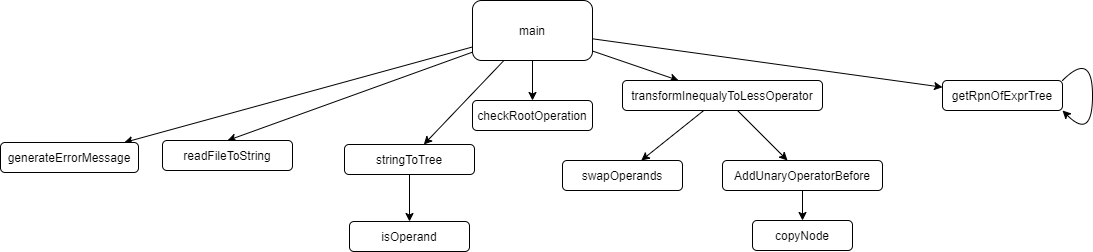


Рис 1. Диаграмма вызовов функций

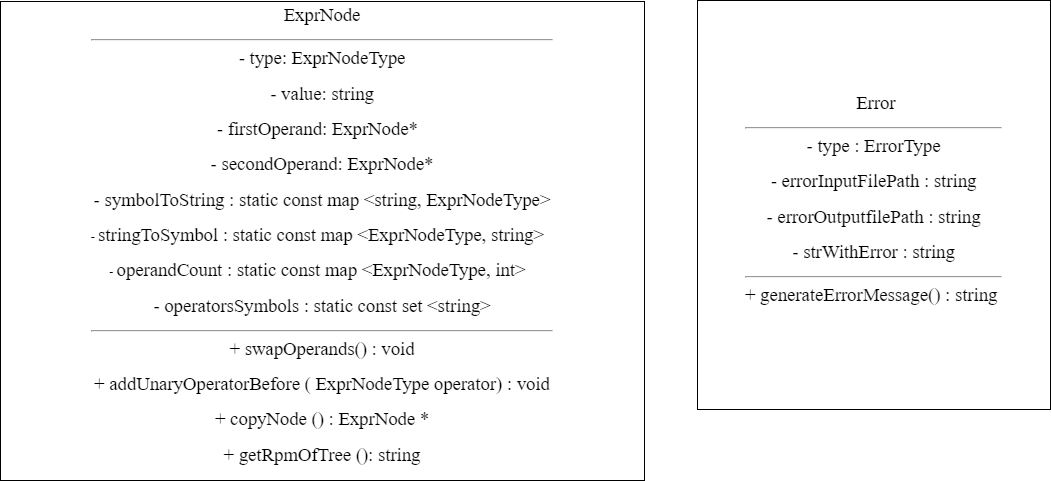


Рис. 2. UML-диаграмма классов

Приложение №2 - Диаграмма потоков данных

