|  |  |
| --- | --- |
| Контролер скобок | |
| Внутренняя спецификация | |
| Студент | Бебахани А. |
| Преподаватель | Матюшечкин Д.С.  Сычев О.А. |
| Сдано |  |

# 1 Общие сведения

Название программы: “BracketChecker2”.

Для функционирования программы необходима операционная система

Windows XP или выше.

Программа написана на языке C++ с использованием его стандартных библиотек.

# 2 Функциональное назначение

Программа считывает текст из переданного файла, содержащего программу на языке C++ и определяет, что каждая открывающая скобка имеет соответствующую закрывающую.

Программа может быть использована в качестве простого и быстрого синтаксического анализатора.

# 3 Описание логической структуры

Работа программы логически разделена на несколько частей:

- считывание потока входных данных во внутренние представление;

- анализ различных встреченных скобок во внутреннем представлении;

- определение корректности расставленных скобок.

Функции, используемые в задаче описаны в приложении А.

Используемые структуры данных описаны в приложении Б.

Диаграмма потоков данных представлена в приложении В.

# 4 Используемые технические средства

Описание требуемых технических средств содержится в Техническом задании в пункте 3.3.

# 

# 5 Вызов и загрузка

Программа запускается из командной строки. Программа принимает путь к файлу в качестве первого аргумента командной строки.

Пример запуска из командной строки:

BracketChecker2.exe test.cpp

# 6 Входные и выходные данные

Формат входных и выходных данных описан в пункте 3.4 технического задания.

# Приложение А

Описание функций

Главная функций программы:

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

Входные данные:

argc - количество переданных аргументов командной строки;

argv[] - аргументы командной строки;

argv[1] - текстовый файл с программой на C++.

Выходные данные:

0 – функция завершилась успешно

1 – произошла ошибка, связанная с выбранным файлом

Алгоритм функции:

1. Считать входные данные.
2. Вызвать функцию проверки правильности расставленных скобок.
3. Вывести результат работы (корректно или с ошибками).

Проверка корректности расставленных скобок:

vector<pair<char, pair<int, int>>> parse\_brackets(const vector<string>& lines);

Входные данные:

lines – вектор, содержащий считанные из файла строки.

Алгоритм функции:

1. Для каждой переданной строки
   1. Считать, что мы находимся вне комментария.
   2. Считать, что мы находимся вне строкового литерала.
   3. Для каждого символа в строке
      1. Если текущий символ и следующий являются начало однострочного комментария
         1. Перейти к следующей строке
      2. Иначе если текущий символ и следующий образуют начало многострочного комментарий
         1. Перейти к месту конца многострочного комментария
      3. Иначе если текущий символ является кавычкой
         1. Пропустить все символы закрывающейся кавычки
      4. Иначе если текущий символ – это открывающаяся скобка
         1. Поместить скобку в стек
      5. Иначе если текущий символ – это закрывающаяся скобка
         1. Если последняя скобка в стеке является парной
            1. Убрать ее из стека
         2. Иначе
            1. Зафиксировать ошибку расстановки скобок

Определяет, является ли переданный символ открывающей скобкой:

bool isOpeningBracket(char ch);

Входные данные:

ch – символ для проверки.

Выходные данные:

Булевский флаг, является ли символ открывающейся скобкой

Алгоритм функции:

1. Сравнить переданный символ ch с каждым элементом списка открывающих скобок ('(', '[', '{')
2. Считать символ ch открывающейся скобкой, если он совпадает с одним из встреченных символов

Определяет, является ли переданный символ закрывающей скобкой (')', ']', '}'):

bool isClosingPair(char open, char close);

Входные данные:

ch – символ для проверки.

Выходные данные:

Булевский флаг, является ли символ закрывающейся скобкой

Алгоритм функции:

1. Сравнить переданный символ ch с каждым элементом списка открывающих скобок (')', ']', '}')
2. Считать символ ch закрывающейся скобкой, если он совпадает с одним из встреченных символов

Проверяет, совпадает ли пара скобок: открывающая и соответствующая ей закрывающая (например, '(' и ')', '[' и ']', '{' и '}'):

bool isMatchingPair(char open, char close);

Входные данные:

open – «открывающий» символ;

open – «закрывающий» символ.

Выходные данные:

Булевский флаг, являются ли символы парными открывающейся и закрывающейся скобкой

Алгоритм функции:

1. Сравнить переданный символ open с каждым элементом списка открывающих скобок ('(', '[', '{').  
 1.1. Если open равен '('  
  1.1.1. Считать, что ожидаемая закрывающая скобка равна ')'  
 1.2. Иначе если open равен '['  
  1.2.1. Считать, что ожидаемая закрывающая скобка равна ']'  
 1.3. Иначе если open равен '{'  
  1.3.1. Считать, что ожидаемая закрывающая скобка равна '}'

2. Сравнить переданный символ close с ожидаемой закрывающей скобкой.  
 2.1. Если close совпадает с ожидаемой закрывающей скобкой  
  2.1.1. Вернуть булевское значение true  
 2.2. Иначе  
  2.2.1. Вернуть булевское значение false

Вывод результата работы:

void print\_result(const string& outputFilename, const vector<pair<char, pair<int, int>>>& errors);

Входные данные:

outputFilename – имя выходного файла;

errors – структура, содержащая скобку, которая ожидалась, и позицию, где она ожидалась.

Алгоритм функции:

1. Открыть выходной файл.
2. Если в ходе анализа не было выявлено ошибок в расстановки скобок
   1. Вывести сообщение о том, что скобки расставлены корректно.
3. Иначе
   1. Для каждой выявленной ошибки
      1. Вывести сообщение о том, какая скобка и на какой позиции она ожидалась.

# Приложение Б

Диаграмма вызов функций

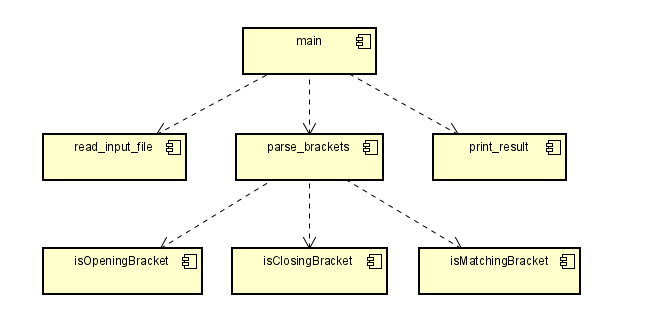


Рис. 1. Диаграмма вызов функций

# Приложение В

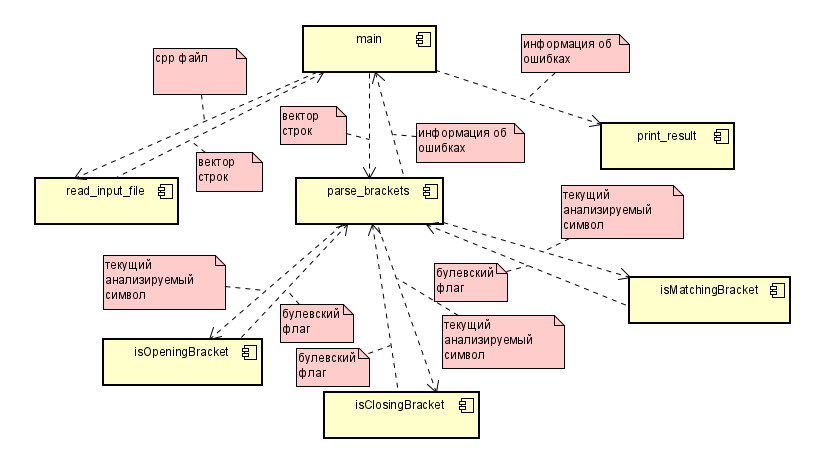
Диаграмма потоков данных 

Рис. 2. Диаграмма потоков данных