|  |  |
| --- | --- |
| Программа для расчёта объёма воды после дождя | |
| Внутренняя спецификация | |
| Студент | Гвоздков С.А. |
| Преподаватель | доц. Сычев О.А. |
| Сдано |  |

1. Назначение

Программа предназначена для осуществления расчета объема воды, который может быть удержан между стен.

1. Описание структур данных

Перечисление класса ErrorType содержит значения:

ManyLinesInInputFile, NotANumber, OutOfRange, NotTxtExtension, FileNotFound, TooManyNumbersInFile, NoNumbers

Структура ErrorInfo содержит поля:

ErrorType type, string detail

1. Описание алгоритмов функций

**Главная функция программы:**

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

Входные данные:

argc – количество переданных аргументов командной строки

argv[] – аргументы командной строки

Выходные данные:

0 – функция завершилась успешно

Алгоритм функции:

1. Считать данные из входного файла (readFromFile)
2. Если есть ошибки, вывести их в консоль и завершить работу программы
3. Вычислить объем воды, который мы можем удержать (calculateWaterVolume)
4. Построить схему стен (drawWallSchema)
5. Записать объем воды и схему стен в выходной файл (writeInFile)

**Функция считывания данных из файла (readFromFile):**

set<ErrorInfo> readFromFile(const string &file\_path, vector<uint32\_t> &numbers);

Входные данные:

file\_path – путь к файлу для чтения

numbers - вектор, в который будут сохранены данные из файла.

Выходные данные:

set<ErrorInfo> - Вектор ошибок, если они есть, иначе пустой вектор.

Алгоритм функции:

1. Получаем расширение файла (getFileExtension)
2. Если расширение не “.txt”, то в вектор ошибок записываем ошибку “NotTxtExtension”
3. Пытаемся открыть файл для чтения.
4. Если файл не открывается, то в вектор ошибок записываем ошибку “FileNotFound”
5. Пытаемся считывать данные
6. Если в файле больше, чем одна строка, то в вектор ошибок записываем ошибку “ManyLinesInInputFile”
7. Разбиваем строку через пробел, для получения входных параметров
   1. Для каждого входного параметра
      1. Если входной параметр не является числом, записываем в массив ошибок этот входной параметр и в вектор ошибок записываем ошибку “NotANumber”
      2. Если число выходит за диапазон [0...4294967295], записываем в массив ошибок этот входной параметр и в вектор ошибок записываем ошибку “OutOfRange”
      3. Добавляем число в вектор numbers
      4. Если чисел в векторе больше 100 чисел, то в вектор ошибок записываем ошибку “TooManyNumbersInFile”
   2. Если чисел в векторе нет, то в вектор ошибок записываем ошибку “NoNumbers”
   3. Возвращаем вектор ошибок

**Функция для получения расширения файла (getFileExtension)** string getFileExtension(const string &filename);

Входные данные:

filename - имя файла

Выходные данные:

string Расширение файла

Алгоритм функции:

1. Нахождение позиции последней точки (.) в переданной строке filename.
2. Проверка, найдена ли точка в строке:
   1. Если точка не найдена (позиция lastDotPos равна string::npos), то расширение файла отсутствует, и функция возвращает пустую строку "".
3. Если точка найдена:
   1. Извлечение подстроки, начиная с позиции после последней точки (т.е. с lastDotPos + 1).
   2. Создание строки extension, добавляя к подстроке символ точки в начале, чтобы получить полное расширение файла.
4. Возврат полученного расширения extension.

**Функция для вычисления объема воды (calculateWaterVolume)**

uint32\_t calculateWaterVolume(const vector<uint32\_t> &wall\_heights, vector<uint32\_t> &water\_heights);

Входные данные:

wall\_heights - вектор высот стен

Выходные данные:

uint\_32t Общее количество воды, удерживаемой между стенами.

water\_heights - вектор для хранения высот воды.

Алгоритм функции:

1. Если в wall\_heights количество стен недостаточное для удержания воды (0…2), возвращаем 0
2. Вычисление максимальных высот слева:
   1. Проходим по всем стенам слева направо.
   2. Для каждой стены сохраняем максимальную высоту, которую встречали до этой стены, и сохраняем ее в max\_left.
3. Вычисление максимальных высот справа:
   1. Проходим по всем стенам справа налево.
   2. Для каждой стены сохраняем максимальную высоту, которую встречали до этой стены, и сохраняем ее в max\_right.
4. Вычисление удерживаемой воды:
   1. Проходим по всем стенам.
   2. Для каждой стены вычисляем высоту воды, которую она может удерживать, используя минимальную из максимальных высот слева и справа, минус высота текущей стены.
   3. Если вычисленная высота воды положительная, добавляем ее к общему количеству воды и сохраняем в water\_heights.
5. Возвращение результата: После завершения итераций возвращается суммарное количество воды, которое может быть задержано между стенами.

**Функция для создания схемы стен (drawWallSchema)**

string drawWallSchema(const vector<uint32\_t> &wall\_heights, vector<uint32\_t> &water\_heights);

Входные данные:

wall\_heights - вектор высот стен.

water\_heights - вектор высот воды.

Выходные данные:

string строка, представляющая схему стен и воды.

Алгоритм функции:

1. Проходим по строкам схемы от максимальной высоты к 1.
   1. Для каждой строки проходим по каждой колонке:
      1. Если текущая строка ниже или на уровне высоты стены, добавляем символ стены (#).
      2. Если текущая строка ниже или на уровне суммы высоты стены и высоты воды, добавляем символ воды (~), затем уменьшаем высоту воды для этой колонки.
      3. В противном случае добавляем пробел ( ).
   2. После каждой колонки добавляем пробел для визуального разделения.
   3. После каждой строки добавляем символ новой строки (\n).
2. Возвращается сформированная строка schema, представляющая схему стен и воды.

**Функция для записи данных в файл (writeInFile)**

void writeInFile(const string &file\_path, uint32\_t water\_volume, const string &wall\_schema);

Входные данные:

file\_path - путь к файлу для записи.

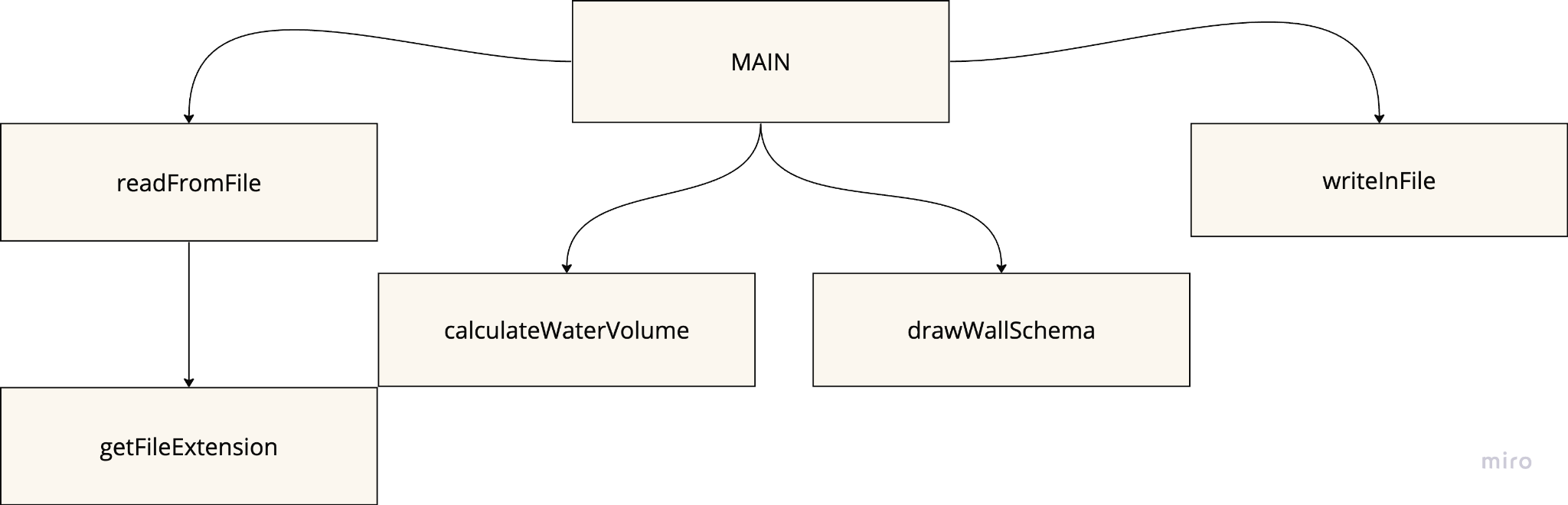
water - количество воды.

walls - рисунок стены и воды.

Алгоритм функции:

1. Открытие файла для записи по указанному пути file\_path.
2. Проверка успешности открытия файла:
   1. Если файл не удалось открыть, выводится сообщение об ошибке и функция завершается.
3. Запись переданных данных в файл:
   1. Для каждого переданного аргумента args производится передача его значения в поток вывода output\_file.
   2. Для этого используется оператор "разворачивания" параметров ..., который разворачивает каждый аргумент из списка args в вызов функции output\_file.
4. Закрытие файла после записи данных.
5. Вывод сообщения об успешном выполнении операции записи в файл.

4. Диаграмма вызовов функций:



5. Диаграмма потоков данных:

