Программа для распаковки строки	
Внутренняя спецификация	
Студент	Николаев А. Д.
Преподаватель	преп. каф. ПОАС Матюшечкин Д.С.
Сдано	
Лабораторная работа №2	

1 Общие сведения

Наименование программы – «Программа для распаковки строки».

Для функционирования программы необходима операционная система Windows 8 или выше.

Программа написана на языке C++.

2 Описание логической структуры программы

2.1 Алгоритм программы

1. Считать входные данные из входного файла.
2. Проверить корректность входных данных.
3. Распаковать строку.
4. Разбить строку на массив строк по 40 символов.
5. Записать выходные данные в выходной файл.

2.2 Декомпозиция программы

Выделенные подпрограммы (функции) описаны в приложении А.

Основные типы и структуры данных программы описаны в приложении Б.

Иерархия вызовов подпрограмм представлена в приложении В.

Диаграмма потоков данных представлена в приложении Г.

Приложение А

Описание функций

Функция: **int main(const int argc, char** argv)**

Обеспечить считывание из файла, вывод в консоль ошибок, если они есть, запись ответа в выходной файл, вызов главной функции, решающей задачу.

Алгоритм работы функции (псевдокод):

Выдать ошибку, если входной файл не указан в аргументах командной строки

Выдать ошибку, если входной файл невозможно открыть

Считать строку из файла...

Распаковать строку...

Если есть ошибка

{

    Распечатать её в консоль ошибок

    Завершить работу программы

}

Разбить распакованную строку на массив строк по длине 40 символов...

Записать массив строк в выходной файл...

Функция: **std::string StringProcessing (std::string_view inputData)**

Распаковать строку.

Алгоритм работы функции (псевдокод):

Выдать ошибку, если размер строки не соответствует разрешенному диапазону

Выдать ошибку, если содержимое строки не корректно…

Разделить строку на модули...

Перевести модули в распакованную строку...

Создать строку, в которую будут распакованы модули

Для каждого модуля...

{

    Добавить к строке символ, содержащийся в модуле в количестве, которое указано в модуле

}

Вернуть распакованную строку

Функция: **std::vector<Unit> DivideStringToChars (std::string_view inputData)**

Разделить строку на модули.

Алгоритм работы функции (псевдокод):

Создать массив, в который будем сохранять модули
Для каждого символа строки:
{
    Если текущий символ – буква
    {
        Добавить в массив модуль с текущим символом и количеством символов равным 1
        Перейти к следующей итерации
    }
    Сформировать модуль и записать его в массив
}
Вернуть получившийся массив юнитов.

Функция: **TermError LateEvaluation (const std::string_view inputData)**

Проверить содержимое строки на корректность.

Алгоритм работы функции:

Для каждого символа строки
{
    Вернуть код ошибки типа "некорректный символ", если найден символ, который не является заглавной латинской буквой или цифрой
    Вернуть код ошибки типа "последняя цифра", если последний символ является цифрой
}
Вернуть код без ошибки

Функция: **std::vector<std::string> DivideString(const std::string_view inputData, const int border)**

Разбить распакованную строку на массив строк по длине 40 символов.

Алгоритм работы функции (псевдокод):

Вернуть пустой массив, если лимит меньше либо равен 0

Создать результирующий массив строк

Для каждой группы символов, длина которой равна лимиту

{

Создать строку из текущей группы символов

Добавить строку в результирующий массив

}

Добавить группу символов в результирующий массив, если она находится конце строки и ее длина меньше лимита

Вернуть результирующий массив

Функция:

Auto FillUnitArray(std::string_view::const_iterator &currentIterator, const std::string_view& inputData, std::vector<Unit> &arrayOfCharacters)

Сформировать модуль и записать его в массив.

Алгоритм работы функции (псевдокод):

Начало числа повторений – текущий символ

Найти первый нецифровой символ – это запоследний символ числа повторений, а также повторяемая буква

Конвертировать строку с числом повторением в число

Если число повторений не входит в разрешенный диапазон – выдать ошибку

Добавить в массив модуль с вычисленным числом повторений и повторяемой буквой

Считать текущим символом следующий символ после повторяемой буквы

Приложение Б

Описание структур данных

Структура **Unit** содержит следующие поля:

int quantityCurrent– количество повторений символа

char Char – повторяющаяся буква.

Структура **TermError** содержит следующие элементы:

Success – нет ошибки.

WrongSymbol – ошибка: неразрешенный символ в строке.

EndNumber – ошибка: в строке последний символ – цифра.

WrongRageOfString – ошибка: размер строки вне разрешенного диапазона.

WrongRepeat – ошибка: число повторений буквы вне разрешенного диапазона.

Структура ExceptionError предназначена для обработки исключений:

```
struct ExceptionError final : std::invalid_argument
{
   using std::invalid_argument::invalid_argument;
   explicit ExceptionError(const TermError code);
private:
   static std::string ErrorManager(const TermError code);
};
```

Приложение В

Иерархия вызовов подпрограмм

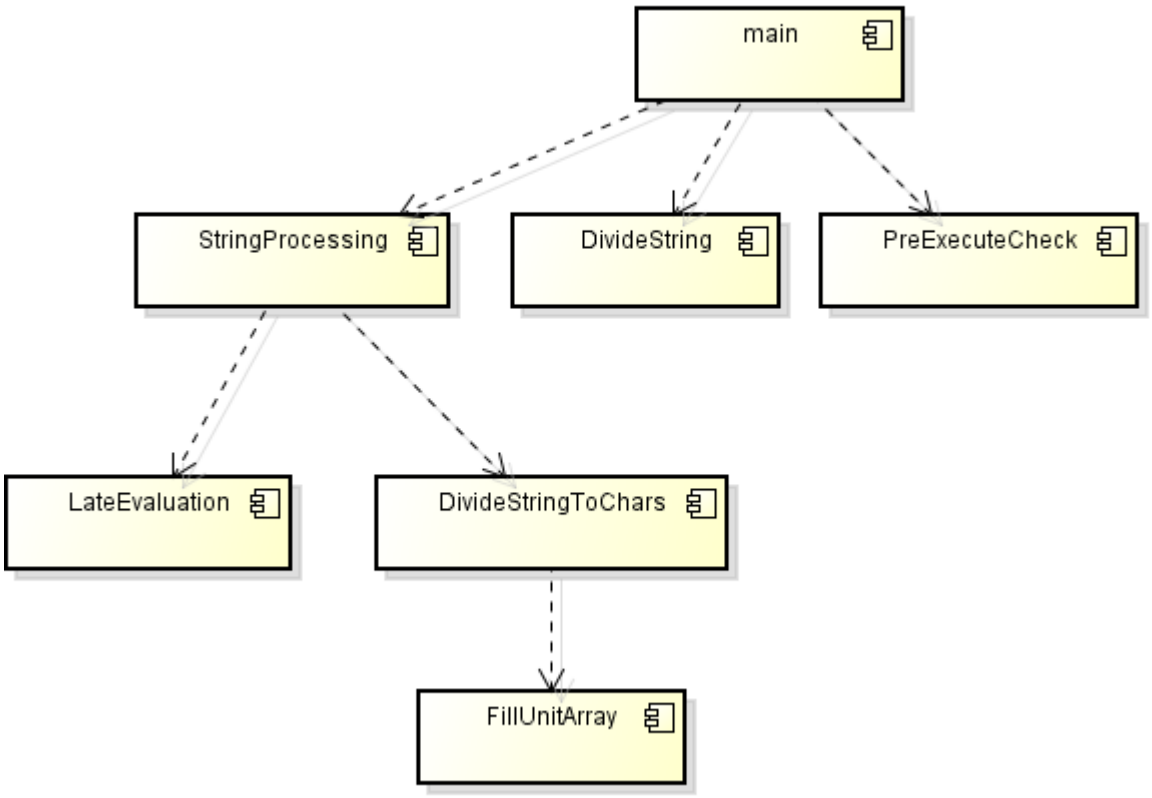

Рис. 1. Иерархия вызовов функций.

Диаграмма потоков данных

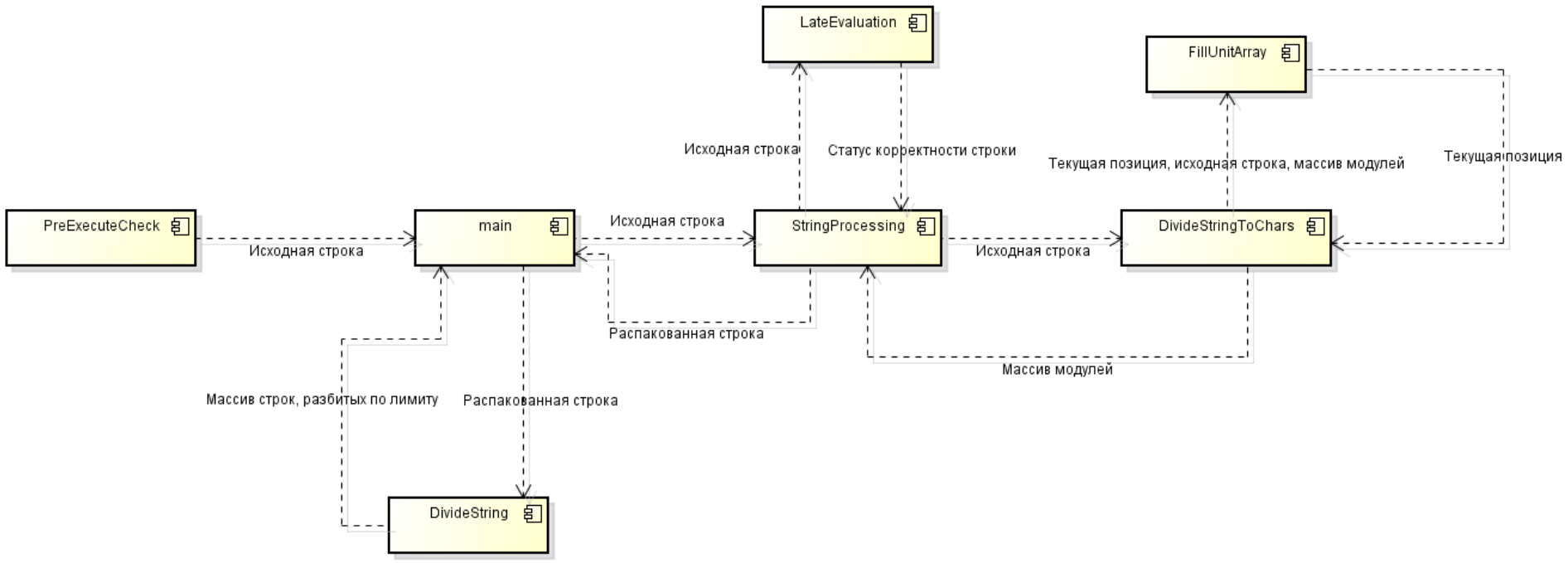

Рис. 2. Диаграмма потоков данных