**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра САПР**

отчет

**по лабораторной работе №2**

**по дисциплине «Программирование»**

|  |  |  |
| --- | --- | --- |
| Студент гр. 1302 |  | Новиков Г.В. |
| Преподаватель |  | Калмычков В.А. |

Санкт-Петербург

2022

Оглавление

[Исходная формулировка 3](#_Toc114193695)

[Контрольный пример 3](#_Toc114193696)

[Формальная постановка задачи 3](#_Toc114193697)

[Формат хранения данных 3](#_Toc114193698)

[Ограничения, обусловленные компьютером 4](#_Toc114193699)

[Макет ввода/вывода 4](#_Toc114193700)

[Алгоритм решения 4](#_Toc114193701)

[Пользовательский тип данных 6](#_Toc114193702)

[Текст программы 6](#_Toc114193703)

[Результат работы программы 9](#_Toc114193704)

[Вывод 9](#_Toc114193705)

# Исходная формулировка

Задан текст, содержащий произвольное количество строк, в которых отдельные слова могут разделяться одним или несколькими пробелами и знаками пунктуации (переносы слов с одной строки на другую не используются). Сформировать новый текст, который является результатом одного из следующих преобразований исходного текста:

Каждое слово, встречающееся более одного раза в тексте, заменить начиная со второго вхождения на обозначение вида S(NN), где S – первый знак слова; NN – порядковый номер в тексте первого вхождения слова.

# Контрольный пример

Текст:

aaaaa aaaaa aaaaa

aaaaa aaaaa aaaaa

Новый текст:

aaaaa

a(1)

a(1)

a(1)

a(1)

a(1)

# Формальная постановка задачи

Дано: текст

Сформировать: новый текст с заменой после второго вхождения

# Формат хранения данных

|  |  |  |
| --- | --- | --- |
| Идентификатор типа | Тип | Назначение |
| file | fstream | Файловые переменные |
| Letters, words, length | char | Символы |
| Number, len, i, start, line-index, index, len, word-count, line\_len, sym | int | Счётчик |
| T[M] | str | Пользовательский тип |
| Marker, mark | char | Маркер |
| append | bool | Дополнительные переменные |
| K, st, end | int | Дополнительные переменные |
| main | int | Основная функция |
| getLen, getNumber | int | Функция обработки |
| setLine, out, printLine, readFromFile, setWord, setLen, printWord, setLetter, setMarker, setNumber, replace\_words | void | Функция обработки |
| is\_sep\_symbol, equals | bool | Функция обработки |
| getLetter, getMarker | char | Функция обработки |

# Ограничения, обусловленные компьютером

Целый тип int имеет размер 4 байта (32 бита). Минимальное значение -2 147 483 648, максимальное значение 2 147 483 647.

Символьный тип char. Минимальное значение -128, максимальное значение 128.

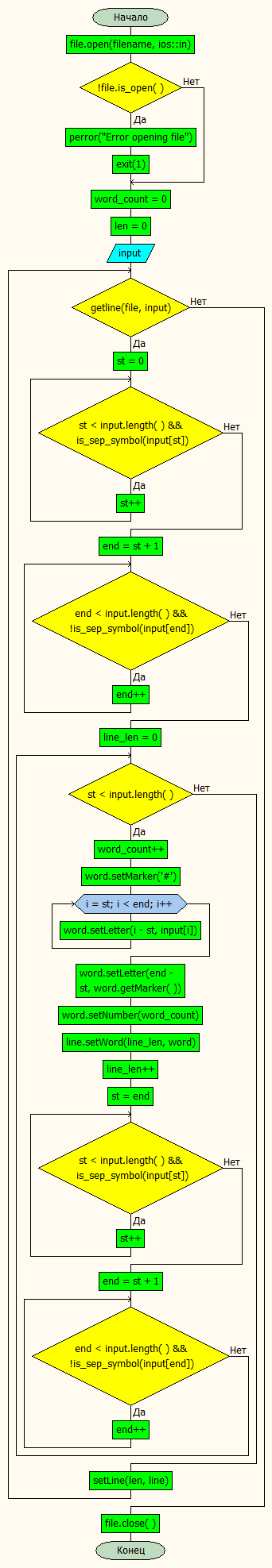
# Макет ввода/вывода

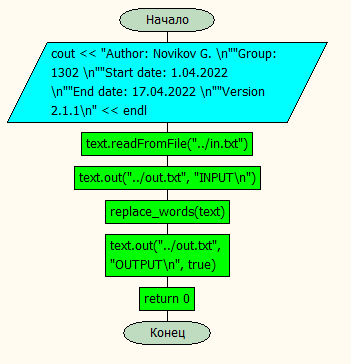
|  |  |
| --- | --- |
| Ввод из файла | file>> |
| Вывод из файла | file<< |

# Алгоритм решения

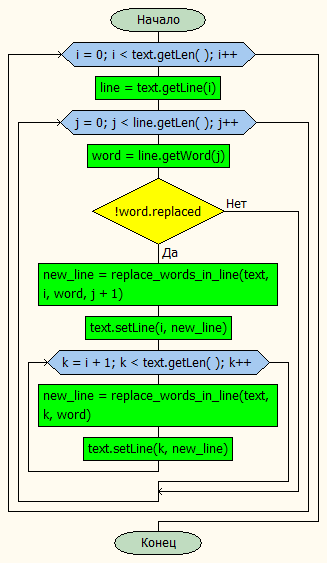
main()

readFromFile ()



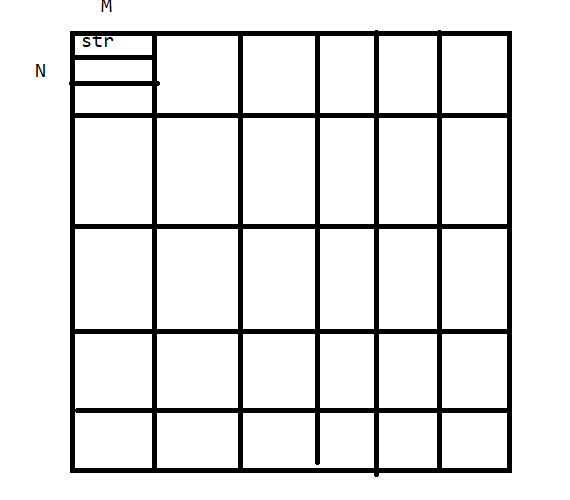


replace\_words ()



# Пользовательский тип данных

Представление текста:



# Текст программы

|  |  |  |
| --- | --- | --- |
| Text.cpp() | Functions.cpp() | Line.cpp () |
| #include "Text.h"  using namespace std;  Line Text::getLine(int index) {  if (index >= len) {  cerr << "Text getLine: Index is bigger than length of text" << endl;  exit(1);  }  return lines[index];  };  void Text::setLine(int index, Line line) {  if (index > len) {  cerr << "Text setLine: Index is bigger than len" << endl;  exit(1);  }  for (int i = 0; i < line.getLen(); i++) {  lines[index].setWord(i, line.getWord(i));  }  if (index == len) len++;  };  void Text::setLen(int length) {  len = length;  };  int Text::getLen() {  return len;  };  void Text::printLine(int index, fstream& file) {  Line line = getLine(index);  for (int i = 0; i < line.getLen(); i++) {  line.printWord(i, file);  if (i != line.getLen() - 1) {  cout << ' ';  file << ' ' << endl;  }  }  cout << endl;  file << endl;  };  void Text::out(string filename, string message, bool append) {  fstream file;  if (append) {  file.open(filename, ios::app);  }  else {  file.open(filename, ios::out);  }  if (!file.is\_open()) {  perror("Error opening file");  exit(1);  }  cout << message;  file << message;  for (int i = 0; i < getLen(); i++) {  printLine(i, file);  }  cout << endl;  file << endl;  file.close();  };  void Text::readFromFile(string filename) {  fstream file;  file.open(filename, ios::in);  if (!file.is\_open()) {  perror("Error opening file");  exit(1);  }  int word\_count = 0;  len = 0;  string input;  while (getline(file, input)) {  Line line;  int st = 0;  while (st < input.length() && is\_sep\_symbol(input[st])) st++;  int end = st + 1;  while (end < input.length() && !is\_sep\_symbol(input[end])) end++;  int line\_len = 0;  while (st < input.length()) {  word\_count++;  Word word;  word.setMarker('#');  for (int i = st; i < end; i++) {  word.setLetter(i - st, input[i]);  }  word.setLetter(end - st, word.getMarker());  word.setNumber(word\_count);  line.setWord(line\_len, word);  line\_len++;  st = end;  while (st < input.length() && is\_sep\_symbol(input[st])) st++;  end = st + 1;  while (end < input.length() && !is\_sep\_symbol(input[end])) end++;  }  setLine(len, line);  }  file.close();  };  bool Text::is\_sep\_symbol(char sym) {  if (sym >= 32 && sym <= 47) {  return true;  }  if (sym >= 58 && sym <= 64) {  return true;  }  if (sym >= 91 && sym <= 96) {  return true;  }  if (sym >= 123 && sym <= 126) {  return true;  }  return false;  }; | #include "functions.h"  using namespace std;  void replace\_words(Text& text) {  for (int i = 0; i < text.getLen(); i++) {  Line line = text.getLine(i);  for (int j = 0; j < line.getLen(); j++) {  Word word = line.getWord(j);  if (!word.replaced) {  Line new\_line = replace\_words\_in\_line(text, i, word, j + 1);  text.setLine(i, new\_line);  for (int k = i + 1; k < text.getLen(); k++) {  Line new\_line = replace\_words\_in\_line(text, k, word);  text.setLine(k, new\_line);  }  }  }  }  };  Line replace\_words\_in\_line(Text text, int line\_index, Word word, int start) {  Line line = text.getLine(line\_index);  for (int j = start; j < line.getLen(); j++) {  if (!line.getWord(j).replaced && line.getWord(j).equals(word)) {  Word new\_word = generate\_new\_word(word, word.getNumber());  line.setWord(j, new\_word);  }  }  return line;  };  Word generate\_new\_word(Word word, int i) {  char first\_letter = word.getLetter(0);  if (first\_letter == word.getMarker()) {  word.replaced = true;  return word;  }  Word new\_word;  new\_word.setMarker(word.getMarker());  new\_word.setLetter(0, first\_letter);  new\_word.setLetter(1, '(');  int num\_len = 1;  if (i <= 9) {  new\_word.setLetter(2, 48 + i);  }  else {  int u = i;  int i\_digits[20];  while (u >= 10) {  i\_digits[num\_len - 1] = u % 10;  u = (int)u / 10;  num\_len++;  }  i\_digits[num\_len - 1] = u;  for (int h = 0; h < num\_len; h++) {  new\_word.setLetter(2 + h, '0' + i\_digits[num\_len - h - 1]);  }  }  new\_word.setLetter(num\_len + 2, ')');  new\_word.setLetter(num\_len + 3, new\_word.getMarker());  new\_word.replaced = true;    return new\_word;  }; | #include "Line.h"  using namespace std;  Word Line::getWord(int index) {  return words[index];  };  void Line::setWord(int index, Word word) {  int k = 0;  if (index > len) {  cerr << "Line setWord: Index is bigger than len" << endl;  exit(1);  }  int i = 0;  while (word.getLetter(i) != word.getMarker()) {  words[index].setLetter(i, word.getLetter(i));  i++;  }  words[index].setMarker(word.getMarker());  words[index].setLetter(i, word.getMarker());  words[index].setNumber(word.getNumber());  words[index].replaced = word.replaced;  if (index == len) len++;  };  int Line::getLen() {  return len;  };  void Line::setLen(char length) {  len = length;  };  void Line::printWord(int index, fstream &file) {  Word word = getWord(index);  int i = 0;  while (word.getLetter(i) != word.getMarker()) {  cout << word.getLetter(i);  file << word.getLetter(i);  i++;  }  }; |
| Functions.h () | Line.h() | Text.h() |
| #include "Text.h"  void replace\_words(Text &text);  Word generate\_new\_word(Word word, int i);  Line replace\_words\_in\_line(Text text, int line\_index, Word word, int start = 0); | #include "Word.h"  class Line {  public:  Word getWord(int index);  void setWord(int index, Word word);  int getLen();  void setLen(char length);  void printWord(int index, fstream &file);  private:  Word words[30];  int len = 0;  }; | #include "Line.h"  class Text {  public:  Line getLine(int index);  void setLine(int index, Line line);  int getLen();  void setLen(int length);  void out(string filename, string message, bool append = 0);  void printLine(int index, fstream& file);  void readFromFile(string filename);  bool is\_sep\_symbol(char sym);  private:  Line lines[100];  int len = 0;  }; |
| main() | Word.h() | Word.cpp() |
| #include "functions.h"  using namespace std;  int main(int argc, char const \*argv[]) {  cout << "Author: Novikov G. \n"  "Group: 1302 \n"  "Start date: 1.04.2022 \n"  "End date: 17.04.2022 \n"  "Version 2.1.1\n" << endl;  Text text;  text.readFromFile("../in.txt");  text.out("../out.txt", "INPUT\n");  replace\_words(text);  text.out("../out.txt", "OUTPUT\n", true);  return 0;  } | #include <iostream>  #include <fstream>  #include <string>  #include <sstream>  using namespace std;  class Word {  public:  char getLetter(int index);  void setLetter(int index, char letter);  char getMarker();  void setMarker(char mark);  int getNumber();  void setNumber(int num);  bool equals(Word word);  bool replaced = false;  private:  char letters[100];  char marker = '#';  int number = -1;  }; | #include "Word.h"  using namespace std;  char Word::getMarker(){  return marker;  };  char Word::getLetter(int index){  return letters[index];  };  void Word::setLetter(int index, char letter){  letters[index] = letter;  };  void Word::setMarker(char mark){  marker = mark;  };  int Word::getNumber() {  return number;  };  void Word::setNumber(int num) {  number = num;  };  bool Word::equals(Word word) {  int i = 0;  while (getLetter(i) != getMarker()) {  if (getLetter(i) != word.getLetter(i)) return false;  i++;  }  if (word.getLetter(i) == word.getMarker()) return true;  return false;  }; |

# Результат работы программы

# Вывод

Работа с вводом и выводом из текстового файла, обработка текста и строк с помощью функций и структур.