**1.1**

#include <iostream>

#include <fstream>

#include <sstream>

using namespace std;

struct Word {

static const int sz = 80;

char letters[sz + 1];

char marker = 0;

Word() {

for (int i = 0; i <= sz; i++) {

letters[i] = '.';

}

letters[sz] = marker;

}

};

int count\_lines(string filename) {

fstream file;

file.open(filename, ios::in);

if (!file.is\_open()) {

perror("Error opening file");

exit(1);

}

string line;

int number\_of\_lines = 0;

while(getline(file, line)) number\_of\_lines++;

file.close();

return number\_of\_lines;

}

int get\_number\_of\_els\_in\_line(string filename, int line\_index = 0) {

fstream file;

file.open(filename, ios::in);

if (!file.is\_open()) {

perror("Error opening file");

exit(1);

}

// find line

string line;

int i = 0;

while (i <= line\_index) {

getline(file, line);

i++;

}

int count = 0;

stringstream ss(line);

string x;

while (getline(ss, x, ' ')) count++;

file.close();

return count;

}

int read\_word\_with\_len(string filename, Word& word, int line\_index = 0) {

int number\_of\_lines = count\_lines(filename);

if (line\_index >= number\_of\_lines) {

cerr << "Error: trying to read non-existing or empty line" << endl;

}

int num\_of\_els = get\_number\_of\_els\_in\_line(filename, line\_index);

if (num\_of\_els < 2) {

cerr << "Error: invalid line" << endl;

}

fstream file;

file.open(filename, ios::in);

if (!file.is\_open()) {

perror("Error opening file");

exit(1);

}

// find line

string line;

int i = 0;

while (i <= line\_index) {

getline(file, line);

i++;

}

stringstream ss(line);

string x;

// read len

getline(ss, x, ' ');

// check if len consists of digits

for (char d : x) {

bool is\_digit = false;

for (char i = '0'; i <= '9'; i++) {

if (d == i) {

is\_digit = true;

break;

}

}

if (!is\_digit) {

cerr << "Error: invalid word length" << endl;

exit(1);

}

}

int len = stoi(x);

// read marker

getline(ss, x, ' ');

if (x.length() != 1) {

cerr << "Error: invalid marker" << endl;

exit(1);

}

word.marker = x.c\_str()[0];

// read word

getline(ss, x);

if (x.length() < len) {

for (int i = 0; i < x.length(); i++) {

word.letters[i] = x[i];

}

len = x.length();

} else {

for (int i = 0; i < len; i++) {

word.letters[i] = x[i];

}

}

word.letters[len] = word.marker;

file.close();

return len;

}

char read\_word\_with\_sep(string filename, Word& word, int line\_index = 0) {

int number\_of\_lines = count\_lines(filename);

if (line\_index >= number\_of\_lines) {

cerr << "Error: trying to read non-existing or empty line" << endl;

}

fstream file;

file.open(filename, ios::in);

if (!file.is\_open()) {

perror("Error opening file");

exit(1);

}

// find line

string line;

int i = 0;

while (i <= line\_index) {

getline(file, line);

i++;

}

// error if the marker is missing

if (line.length() < 2) {

cerr << "Invalid line" << endl;

exit(1);

}

// read separator and marker

char sep = line[0];

word.marker = line[1];

// find start and end separators

int start;

int end;

i = 2;

while (i < line.length() && line[i] != sep) i++;

start = i;

i++;

while (i < line.length() && line[i] != sep) i++;

end = i;

// read word from start + 1 to end - 1

if (start < line.length() - 1) {

for (i = start + 1; i < end; i++) {

word.letters[i - start - 1] = line[i];

}

}

// terminating character

word.letters[i - start - 1] = word.marker;

file.close();

return sep;

}

void read\_substrs(string filename, Word subs[][2], int len, char marker) {

fstream file;

file.open(filename, ios::in);

if (!file.is\_open()) {

perror("Error opening file");

exit(1);

}

string line;

for (int i = 0; i < len; i++) {

int els\_count = get\_number\_of\_els\_in\_line(filename, i);

if(els\_count < 2) {

cerr << "Too little substrings in line " << i + 1 << endl;

exit(1);

}

Word sub1;

Word sub2;

sub1.marker = marker;

sub2.marker = marker;

sub1.letters[sub1.sz] = marker;

sub2.letters[sub2.sz] = marker;

getline(file, line);

int k = 0;

while (line[k] != ' ') {

sub1.letters[k] = line[k];

k++;

}

sub1.letters[k] = marker;

int l = k + 1;

while (line[l] && line[l] != ' ') {

sub2.letters[l - k - 1] = line[l];

l++;

}

sub2.letters[l - k - 1] = marker;

subs[i][0] = sub1;

subs[i][1] = sub2;

}

file.close();

}

int find\_substr(const Word &word, const Word &sub, int word\_len, int sub\_len, int search\_start = 0) {

if(sub\_len == 0) return -1;

bool is\_equal = true;

for(int i = search\_start; i <= word\_len - sub\_len; i++) {

is\_equal = true;

for(int j = 0; j < sub\_len; j++) {

if(sub.letters[j] != word.letters[i + j]) {

is\_equal = false;

break;

}

}

if(is\_equal) {

return i;

}

}

return -1;

}

void replace(Word &word, const Word &removed, const Word &inserted) {

int word\_len = 0;

int removed\_len = 0;

int inserted\_len = 0;

while (word.letters[word\_len] != word.marker) {

word\_len++;

}

while (removed.letters[removed\_len] != removed.marker) {

removed\_len++;

}

while (inserted.letters[inserted\_len] != inserted.marker) {

inserted\_len++;

}

int i = find\_substr(word, removed, word\_len, removed\_len);

while(i != -1) {

int shift = inserted\_len - removed\_len;

if(word\_len + shift > word.sz) {

cerr << "Error: word length after replacement is bigger than max size" << endl;

exit(1);

}

if(shift > 0) {

for(int k = word\_len - 1; k >= i; k--) {

word.letters[k + shift] = word.letters[k];

}

} else if(shift < 0) {

for(int k = i; k < word\_len + shift; k++) {

word.letters[k] = word.letters[k - shift];

}

}

for (int k = i; k < i + inserted\_len; k++) {

word.letters[k] = inserted.letters[k - i];

}

word.letters[word\_len + shift] = word.marker;

word\_len += shift;

i = find\_substr(word, removed, word\_len, removed\_len, i + inserted\_len);

}

}

void out\_init\_sep(string filename, Word word, char sep, bool app = false) {

fstream file;

if (app) {

file.open(filename, ios::app);

} else {

file.open(filename, ios::out);

}

if (!file.is\_open()) {

perror("Error opening file");

exit(1);

}

if (!app) {

file << "INPUT" << endl;

cout << "INPUT" << endl;

} else {

file << endl;

cout << endl;

}

file << "Separator: " << sep << endl;

file << "Marker: " << word.marker << endl;

cout << "Separator: " << sep << endl;

cout << "Marker: " << word.marker << endl;

int last\_marker;

for (int i = 0; i < word.sz; i++) {

if (word.letters[i] == word.marker) {

last\_marker = i;

}

}

file << "Word: ";

cout << "Word: ";

for (int i = 0; i < last\_marker; i++) {

file << word.letters[i];

cout << word.letters[i];

}

file << endl;

cout << endl;

file.close();

}

void out\_init\_len(string filename, Word word, int len, bool app = false) {

fstream file;

if (app) {

file.open(filename, ios::app);

} else {

file.open(filename, ios::out);

}

if (!file.is\_open()) {

perror("Error opening file");

exit(1);

}

if (!app) {

file << "INPUT" << endl;

cout << "INPUT" << endl;

} else {

file << endl;

cout << endl;

}

file << "Length: " << len << endl;

file << "Marker: " << word.marker << endl;

file << "Word: ";

cout << "Length: " << len << endl;

cout << "Marker: " << word.marker << endl;

cout << "Word: ";

for (int i = 0; i < len; i++) {

file << word.letters[i];

cout << word.letters[i];

}

file << endl;

cout << endl;

file.close();

}

void out\_res(string filename, Word word, bool app\_res = false) {

fstream file;

file.open(filename, ios::app);

if (!file.is\_open()) {

perror("Error opening file");

exit(1);

}

if (!app\_res) {

file << endl << "OUTPUT" << endl;

cout << endl << "OUTPUT" << endl;

} else {

file << endl;

cout << endl;

}

file << "Word: ";

cout << "Word: ";

int i = 0;

while (word.letters[i] != word.marker) {

file << word.letters[i];

cout << word.letters[i];

i++;

}

file << endl;

cout << endl;

file.close();

}

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

cout << "Author: Novikov G. \n"

"Group: 1302 \n"

"Start date: 14.02.2022 \n"

"End date: 15.02.2022 \n"

"Version 1.1.1\n" << endl;

Word word;

char mode = 's';

if (mode == 's') {

char sep = read\_word\_with\_sep("in.txt", word);

out\_init\_sep("out.txt", word, sep);

} else if (mode == 'l') {

int len = read\_word\_with\_len("in.txt", word);

out\_init\_len("out.txt", word, len);

} else exit(1);

int subs\_count = count\_lines("substrings.txt");

Word subs[subs\_count][2];

read\_substrs("substrings.txt", subs, subs\_count, word.marker);

for (int i = 0; i < subs\_count; i++) {

replace(word, subs[i][0], subs[i][1]);

}

out\_res("out.txt", word);

return 0;

}#include <iostream>

#include <fstream>

#include <sstream>

using namespace std;

struct Word {

static const int sz = 80;

char letters[sz + 1];

char marker = 0;

Word() {

for (int i = 0; i <= sz; i++) {

letters[i] = '.';

}

letters[sz] = marker;

}

};

int count\_lines(string filename) {

fstream file;

file.open(filename, ios::in);

if (!file.is\_open()) {

perror("Error opening file");

exit(1);

}

string line;

int number\_of\_lines = 0;

while(getline(file, line)) number\_of\_lines++;

file.close();

return number\_of\_lines;

}

int get\_number\_of\_els\_in\_line(string filename, int line\_index = 0) {

fstream file;

file.open(filename, ios::in);

if (!file.is\_open()) {

perror("Error opening file");

exit(1);

}

// find line

string line;

int i = 0;

while (i <= line\_index) {

getline(file, line);

i++;

}

int count = 0;

stringstream ss(line);

string x;

while (getline(ss, x, ' ')) count++;

file.close();

return count;

}

int read\_word\_with\_len(string filename, Word& word, int line\_index = 0) {

int number\_of\_lines = count\_lines(filename);

if (line\_index >= number\_of\_lines) {

cerr << "Error: trying to read non-existing or empty line" << endl;

}

int num\_of\_els = get\_number\_of\_els\_in\_line(filename, line\_index);

if (num\_of\_els < 2) {

cerr << "Error: invalid line" << endl;

}

fstream file;

file.open(filename, ios::in);

if (!file.is\_open()) {

perror("Error opening file");

exit(1);

}

// find line

string line;

int i = 0;

while (i <= line\_index) {

getline(file, line);

i++;

}

stringstream ss(line);

string x;

// read len

getline(ss, x, ' ');

// check if len consists of digits

for (char d : x) {

bool is\_digit = false;

for (char i = '0'; i <= '9'; i++) {

if (d == i) {

is\_digit = true;

break;

}

}

if (!is\_digit) {

cerr << "Error: invalid word length" << endl;

exit(1);

}

}

int len = stoi(x);

// read marker

getline(ss, x, ' ');

if (x.length() != 1) {

cerr << "Error: invalid marker" << endl;

exit(1);

}

word.marker = x.c\_str()[0];

// read word

getline(ss, x);

if (x.length() < len) {

for (int i = 0; i < x.length(); i++) {

word.letters[i] = x[i];

}

len = x.length();

} else {

for (int i = 0; i < len; i++) {

word.letters[i] = x[i];

}

}

word.letters[len] = word.marker;

file.close();

return len;

}

char read\_word\_with\_sep(string filename, Word& word, int line\_index = 0) {

int number\_of\_lines = count\_lines(filename);

if (line\_index >= number\_of\_lines) {

cerr << "Error: trying to read non-existing or empty line" << endl;

}

fstream file;

file.open(filename, ios::in);

if (!file.is\_open()) {

perror("Error opening file");

exit(1);

}

// find line

string line;

int i = 0;

while (i <= line\_index) {

getline(file, line);

i++;

}

// error if the marker is missing

if (line.length() < 2) {

cerr << "Invalid line" << endl;

exit(1);

}

// read separator and marker

char sep = line[0];

word.marker = line[1];

// find start and end separators

int start;

int end;

i = 2;

while (i < line.length() && line[i] != sep) i++;

start = i;

i++;

while (i < line.length() && line[i] != sep) i++;

end = i;

// read word from start + 1 to end - 1

if (start < line.length() - 1) {

for (i = start + 1; i < end; i++) {

word.letters[i - start - 1] = line[i];

}

}

// terminating character

word.letters[i - start - 1] = word.marker;

file.close();

return sep;

}

void read\_substrs(string filename, Word subs[][2], int len, char marker) {

fstream file;

file.open(filename, ios::in);

if (!file.is\_open()) {

perror("Error opening file");

exit(1);

}

string line;

for (int i = 0; i < len; i++) {

int els\_count = get\_number\_of\_els\_in\_line(filename, i);

if(els\_count < 2) {

cerr << "Too little substrings in line " << i + 1 << endl;

exit(1);

}

Word sub1;

Word sub2;

sub1.marker = marker;

sub2.marker = marker;

sub1.letters[sub1.sz] = marker;

sub2.letters[sub2.sz] = marker;

getline(file, line);

int k = 0;

while (line[k] != ' ') {

sub1.letters[k] = line[k];

k++;

}

sub1.letters[k] = marker;

int l = k + 1;

while (line[l] && line[l] != ' ') {

sub2.letters[l - k - 1] = line[l];

l++;

}

sub2.letters[l - k - 1] = marker;

subs[i][0] = sub1;

subs[i][1] = sub2;

}

file.close();

}

int find\_substr(const Word &word, const Word &sub, int word\_len, int sub\_len, int search\_start = 0) {

if(sub\_len == 0) return -1;

bool is\_equal = true;

for(int i = search\_start; i <= word\_len - sub\_len; i++) {

is\_equal = true;

for(int j = 0; j < sub\_len; j++) {

if(sub.letters[j] != word.letters[i + j]) {

is\_equal = false;

break;

}

}

if(is\_equal) {

return i;

}

}

return -1;

}

void replace(Word &word, const Word &removed, const Word &inserted) {

int word\_len = 0;

int removed\_len = 0;

int inserted\_len = 0;

while (word.letters[word\_len] != word.marker) {

word\_len++;

}

while (removed.letters[removed\_len] != removed.marker) {

removed\_len++;

}

while (inserted.letters[inserted\_len] != inserted.marker) {

inserted\_len++;

}

int i = find\_substr(word, removed, word\_len, removed\_len);

while(i != -1) {

int shift = inserted\_len - removed\_len;

if(word\_len + shift > word.sz) {

cerr << "Error: word length after replacement is bigger than max size" << endl;

exit(1);

}

if(shift > 0) {

for(int k = word\_len - 1; k >= i; k--) {

word.letters[k + shift] = word.letters[k];

}

} else if(shift < 0) {

for(int k = i; k < word\_len + shift; k++) {

word.letters[k] = word.letters[k - shift];

}

}

for (int k = i; k < i + inserted\_len; k++) {

word.letters[k] = inserted.letters[k - i];

}

word.letters[word\_len + shift] = word.marker;

word\_len += shift;

i = find\_substr(word, removed, word\_len, removed\_len, i + inserted\_len);

}

}

void out\_init\_sep(string filename, Word word, char sep, bool app = false) {

fstream file;

if (app) {

file.open(filename, ios::app);

} else {

file.open(filename, ios::out);

}

if (!file.is\_open()) {

perror("Error opening file");

exit(1);

}

if (!app) {

file << "INPUT" << endl;

cout << "INPUT" << endl;

} else {

file << endl;

cout << endl;

}

file << "Separator: " << sep << endl;

file << "Marker: " << word.marker << endl;

cout << "Separator: " << sep << endl;

cout << "Marker: " << word.marker << endl;

int last\_marker;

for (int i = 0; i < word.sz; i++) {

if (word.letters[i] == word.marker) {

last\_marker = i;

}

}

file << "Word: ";

cout << "Word: ";

for (int i = 0; i < last\_marker; i++) {

file << word.letters[i];

cout << word.letters[i];

}

file << endl;

cout << endl;

file.close();

}

void out\_init\_len(string filename, Word word, int len, bool app = false) {

fstream file;

if (app) {

file.open(filename, ios::app);

} else {

file.open(filename, ios::out);

}

if (!file.is\_open()) {

perror("Error opening file");

exit(1);

}

if (!app) {

file << "INPUT" << endl;

cout << "INPUT" << endl;

} else {

file << endl;

cout << endl;

}

file << "Length: " << len << endl;

file << "Marker: " << word.marker << endl;

file << "Word: ";

cout << "Length: " << len << endl;

cout << "Marker: " << word.marker << endl;

cout << "Word: ";

for (int i = 0; i < len; i++) {

file << word.letters[i];

cout << word.letters[i];

}

file << endl;

cout << endl;

file.close();

}

void out\_res(string filename, Word word, bool app\_res = false) {

fstream file;

file.open(filename, ios::app);

if (!file.is\_open()) {

perror("Error opening file");

exit(1);

}

if (!app\_res) {

file << endl << "OUTPUT" << endl;

cout << endl << "OUTPUT" << endl;

} else {

file << endl;

cout << endl;

}

file << "Word: ";

cout << "Word: ";

int i = 0;

while (word.letters[i] != word.marker) {

file << word.letters[i];

cout << word.letters[i];

i++;

}

file << endl;

cout << endl;

file.close();

}

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

cout << "Author: Novikov G. \n"

"Group: 1302 \n"

"Start date: 14.02.2022 \n"

"End date: 15.02.2022 \n"

"Version 1.1.1\n" << endl;

Word word;

char mode = 's';

if (mode == 's') {

char sep = read\_word\_with\_sep("in.txt", word);

out\_init\_sep("out.txt", word, sep);

} else if (mode == 'l') {

int len = read\_word\_with\_len("in.txt", word);

out\_init\_len("out.txt", word, len);

} else exit(1);

int subs\_count = count\_lines("substrings.txt");

Word subs[subs\_count][2];

read\_substrs("substrings.txt", subs, subs\_count, word.marker);

for (int i = 0; i < subs\_count; i++) {

replace(word, subs[i][0], subs[i][1]);

}

out\_res("out.txt", word);

return 0;

}