Тампио Илья Сергеевич 3530904\10002, Индивидуальное №4, Вариант №5

Описание:

Написать функцию, формирующую новую строку из символов двух исходных строк, которые НЕ являются для них общими.

Функция должна иметь вид

**char \* function(char \*destination, const char \*source, …)**

destination – строка, в которую помещается результат,

source – исходная строка (строки),

возвращаемое значение – указатель на строку результата.

Требования к реализации функции:

Строка destination должна быть передана в функцию извне, чтобы ***избежать*** выделения динамической памяти внутри функции (по примеру стандартных функций из <cstring>). НО *Для исключения повторных проходов по строкам, можно использовать дополнительную память.*

При работе со строками в стиле C не следует передавать в функцию размер строки или

вычислять его внутри функции специально. Все строки должны оканчиваться символом ‘\0’ и нужно работать, учитывая это правило.

Максимальную длину строки (строк) и сами символы строки (строк) надо вводить из файла. Данные для всех тестов могут быть в одном файле. Если в задаче требуется формирование строки-результата, то ее надо вывести в файл-результата.

Строки в стиле С разместить в динамической памяти, максимальный размер строки-результата определить на основе введенных данных.

Результаты всех выполняемых тестов нужно вывести в один файл.

Детальные требования и тестплан:

|  |  |  |  |
| --- | --- | --- | --- |
| Требование | Описания | Данные | Результат |
| 1. Входные данные | | | |
| 1.1 Формат | Данные в файле должны быть организованы следующим образом:  [длина первой строки] [длина второй строки] [первая строка] [вторая строка] |  |  |
| 1.2 Максимальный размер строки результата | За максимальный размер строки результата можно взять наименьшее длин входных строк и числа допустимых символов (по ASCII – 256, включая место под \0) |  |  |
| 1.3 Типы данных | Размер массива – натуральное число, иначе исключение «Неверно задан размер массива, завершение программы…» | 0 А  abc | исключение «Неверно задан размер массива, завершение программы…» |
| 1.4 Ошибка открытия файла | Если файл недоступен или его нет, исключение «Ошибка открытия файла, завершение программы…» |  | Исключение «Ошибка открытия файла» |
| 1. Вывод | | | |
| 2.1 Вывод | Вывод должен осуществляться в файл output.txt |  |  |
| 2.2 Ошибка открытия файла | Если файл недоступен – исключение «Ошибка открытия файла, завершение программы…». (Если файла нет, то программа должна его создать) |  | исключение «Ошибка открытия файла, завершение программы…» |
| 1. Требования к работе программы | | | |
| 3.1 Общие требования к конечной строке | В конечной строке не должно содержаться дубликатов, при работе программы символы разных регистров считаются за разные символы. |  |  |
| 3.2 Требования к размеру конечной строки | Исходя из требования 3.1, размер массива не может превышать суммы длин строк или количество всех возможных символов, смотря что меньше. |  |  |
| 1. Тесты | | | |
| 4.1 |  | 3  abc  def | abcdef |
| 4.2 |  | 6  abcdef  abc | def |
| 4.3 |  | 6  abcdef  def | abc |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Код:

//вар5 тампио 3530904/10002

#include <iostream>

#include <fstream>

#define IFNAME "input.txt"

#define OFNAME "output.txt"

bool aInB(const char a, const char\* b) {

for (int i = 0; b[i] != '\0'; i++) {

if (b[i] == a) {

return true;

}

}

return false;

}

char\* getDiff(char\* destination, const char\* source\_a, const char\* source\_b) {

// O(n^n) worst (case: search a, b, c in deffffff... ...fffabc)

// optimization: sort and leave only unique - but it uses more memory.

int j = 0;

for (int i = 0; source\_a[i] != '\0'; i++) {

if (!aInB(source\_a[i], destination) && !aInB(source\_a[i], source\_b)) {

if (destination[j] == '\0') {

throw "Недостаточно места в конечной строке";

}

destination[j] = source\_a[i];

j++;

}

}

for (int i = 0; source\_a[i] != '\0'; i++) {

if (!aInB(source\_b[i], destination) && !aInB(source\_b[i], source\_a)) {

if (destination[j] == '\0') {

throw "Недостаточно места в конечной строке";

}

destination[j] = source\_b[i];

j++;

}

}

destination[j] = '\0';

return destination;

}

int main()

{

setlocale(LC\_ALL, "RUSSIAN");

std::cout << "\nИз файла...\n";

std::ifstream fin(IFNAME);

std::ofstream fout(OFNAME);

if (!fin.good() || !fout.good()) {

std::cerr << "Ошибка открытия файла. Завершение программы...\n";

return 1;

}

while (!fin.eof()) {

int arraySizeA = -1;

fin >> arraySizeA;

if (arraySizeA <= 0 || !fin.good()) {

std::cerr << "Размер массива1 задан неверно. Завершение программы...\n";

return 1;

}

int arraySizeB = -1;

fin >> arraySizeB;

if (arraySizeB <= 0 || !fin.good()) {

std::cerr << "Размер массива2 задан неверно. Завершение программы...\n";

return 1;

}

char\* charArrayA = new char[arraySizeA + 1];

char\* charArrayB = new char[arraySizeB + 1];

if (fin.eof()) {

std::cerr << "Неожиданный конец файла при чтении массива1. Завершение программы...\n";

return 1;

}

fin >> charArrayA;

if (fin.eof()) {

std::cerr << "Неожиданный конец файла при чтении массива2. Завершение программы...\n";

return 1;

}

fin >> charArrayB;

int arraySizeSum = arraySizeA + arraySizeB;

const int ASCII\_SIZE = 256;

int smallestSize = (arraySizeSum < ASCII\_SIZE) ? arraySizeSum : ASCII\_SIZE;

char\* destinationCharArray = new char[smallestSize + 1];

destinationCharArray[smallestSize] = '\0';

try {

getDiff(destinationCharArray, charArrayA, charArrayB);

}

catch (const char \* error){

std::cerr << error;

return 1;

}

fout << destinationCharArray << '\n';

// LOG

std::clog << "=============================\n";

std::clog << charArrayA << '\n';

std::clog << charArrayB << '\n';

std::clog << destinationCharArray << '\n';

delete[] charArrayA;

delete[] charArrayB;

delete[] destinationCharArray;

}

return 0;

}