|  |  |
| --- | --- |
| C:\Documents and Settings\admin\Рабочий стол\МИРЭА_ЭМБЛЕМА_приказ.JPG | |
| МИНОБРАНАУКИ РОССИИ | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  «**МОСКОВСКИЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»**  **МИРЭА** | |
| Институт информационных технологий  Кафедра математического обеспечения и стандартизации информационных технологий | |
| **Отчет по лабораторной работе №3**  **Тема работы «СТРОКИ»** | |
| Выполнил: |  |
| Студент группы ИКБО-07-17  Отчет принял:  Преподаватель кафедры МОСИТ | Акжигитов Р. Р.  Миронов Антон Николаевич |
| Москва  2018 | |

Оглавление

[Задание 1 3](#_Toc510300484)

[Условия задания 3](#_Toc510300485)

[Декомпозиция 3](#_Toc510300486)

[Реализация функций 3](#_Toc510300487)

[Задание 2 3](#_Toc510300488)

[Условия задания 3](#_Toc510300489)

[Декомпозиция 4](#_Toc510300490)

[Реализация функций 4](#_Toc510300491)

[Функции ввода\вывода массива\значений 5](#_Toc510300492)

[Кодирование алгоритма программы 6](#_Toc510300493)

[Вывод 11](#_Toc510300494)

# Задание 1

# Условия задания

Дан текст, хранящийся в строковой переменной, удалить все гласные буквы.

# Декомпозиция

1. Объявить массив из гласных букв обоих алфавитов.
2. Прохождение по строке, если буква найдена в строке гласных, то не добавляем в новую строку, если она не найдена (то есть она не гласная, так как в строке гласных перечислены только гласные), то добавляем символ в строке, которую одна из функций вернет.

# Реализация функций

char\* vowels = "уеыаоэяиюёeyuioa";

// получение длины строки

int getLength(char\* str) {

int length = 0;

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

length++;

return length;

}

// поиск символа в строке

int indexOf(char ch, char\* string) {

for (int i = 0; i < getLength(string); i++)

if (string[i] == ch) return i;

return -1;

}

// индекс строки в массиве

int indexOf(char\* str, char\*\* strArr, int size) {

for (int i = 0; i < size; i++)

if (!strcmp(str, strArr[i]))

return i;

return -1;

}

// удаление гласных

char\* deleteVowels(char\* str, char\* vw = vowels) {

//char\* temp = (char\*)malloc(sizeof(char) \* maxLength);

char\* temp = new char[getLength(str)];

int cx = 0;

for (int i = 0; i < getLength(str); i++)

if (indexOf(str[i], vw) < 0)

temp[cx++] = str[i];

temp[cx] = '\0';

return temp;

}

# Задание 2

# Условия задания

Дано предложение, составленное из слов и групп цифр (считать тоже словами), определяющих целые числа. Слова разделены запятой или одним или несколькими пробелами. Удалить из него слова, которые встретились там более одного раза и сформировать массив из чисел, встретившихся в тексте.

# Декомпозиция

1. Отделения слова от строки
2. Если слова является целым числом, то добавить в массив чисел.
3. Использовать словарь для определения частоты встречи слова (повторяющихся слов).
4. Если ключ слова уже есть в словаре, то увеличить его значение на 1, если нет добавить ключ.
5. Если ключ слова не найден в словаре, то добавить в массив уникальных слов для будущей пересоздания строки.
6. Во время разбора строки на слова также добавлять в отдельный массив разделители между словами, так как они в пересозданной строке должны будут остаться.
7. Функция пересоздания строки, которая получает массив слов без повторений, с пустыми местами, вместо повторяющихся слов и массив разделителей между словами.

# Реализация функций

// вычленение слова из строки

char\* getWord(char\* str, char\*& newStr, char\*\*& delims, int& delimsSize, char\* delim = " ") {

char\* word = new char[strlen(str)];

delims = (char\*\*)realloc(delims, sizeof(char\*) \* ++delimsSize);

//delims[delimsSize - 1] = (char\*)realloc(delims[delimsSize - 1], sizeof(char) \* 1);

delims[delimsSize - 1] = new char[1];

int cx = 0;

int i = 0;

while (str[i] != '\0' && indexOf(str[i], delim) >= 0) {

delims[delimsSize - 1] = (char\*)realloc(delims[delimsSize - 1], sizeof(char) \* (i + 1));

delims[delimsSize - 1][i] = str[i];

i++;

}

delims[delimsSize - 1][i] = '\0';

while (str[i] != '\0' && indexOf(str[i], delim) < 0) {

word[cx++] = str[i++];

}

newStr = new char[strlen(str) - cx];

word[cx] = '\0';

cx = 0;

while (str[i] != '\0') {

newStr[cx++] = str[i++];

}

newStr[cx] = '\0';

return word;

}

// проверка число ли это

int isNumber(char\* str) {

if (str == NULL || strlen(str) == 0) return 0;

for (int i = 0; i < strlen(str); i++)

if (str[i] - '0' < 0 || str[i] - '0' > 9) return 0;

return 1;

}

// ввод числа с проверкой на диапазон

int input(int min, int max) {

char temp[256];

char\* cArray = NULL;

while (!isNumber(cArray) || stoi(cArray) > max || stoi(cArray) < min) {

cout << "Введите целое число от " << min << " до " << max << ": ";

cin.getline(temp, 256);

cArray = temp;

}

return stoi(cArray);

}

// пересборка строки с массивом удаленных слов

char\* rebuildString(char\*\* words, int wordsSize, char\*\* delims, int delimsSize) {

char\* result = new char[0];

int cx = 0;

int length = 0;

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

length = length + strlen(delims[i]) + strlen(words[i]);

result = (char\*)realloc(result, sizeof(char) \* length);

for (int j = 0; j < strlen(delims[i]); j++)

result[cx++] = delims[i][j];

for (int j = 0; j < strlen(words[i]); j++)

result[cx++] = words[i][j];

}

result[length] = '\0';

return result;

}

# Функции ввода\вывода массива\значений

// ввод массива символов

char\* getCharArray(int length, const char delim = '\n') {

char\* temp = new char[length];

int i = 0;

for (; i < length; i++) {

temp[i] = cin.get();

if (temp[i] == delim) break;

}

temp[i] = '\0';

return temp;

}

// добавление элемента в массив

void append(int\*& arr, int& size, int number) {

arr = (int\*)realloc(arr, sizeof(int) \* ++size);

arr[size - 1] = number;

}

// печать массива целых чисел

void print(int\* arr, int size, char\* delim = ", ") {

if (arr == NULL) {

cout << "Массив пуст...";

return;

}

cout << "[";

for (int i = 0; i < size - 1 && size > 1; i++)

cout << arr[i] << delim;

cout << arr[size - 1] << "]";

}

// печать массива строк

void print(char\*\* arr, int size, char\* delim = ", ") {

if (arr == NULL) {

cout << "Массив пуст...";

return;

}

cout << "[";

for (int i = 0; i < size - 1 && size > 1; i++)

cout << "\"" << arr[i] << "\"" << delim;

cout << "\"" << arr[size - 1] << "\"]";

}

# Кодирование алгоритма программы

#pragma once

#define \_CRT\_SECURE\_NO\_WARNINGS

#include "stdlib.h"

#include "stdio.h"

#include "iostream"

#include "string"

#include "Windows.h"

#include "ctype.h"

using namespace std;

char\* vowels = "уеыаоэяиюёeyuioa";

const int maxLength = 100;

void russianOutput(char\* str) {

setlocale(LC\_ALL, ".866");

cout << str;

setlocale(LC\_ALL, "RUS");

}

// получение длины строки

int getLength(char\* str) {

int length = 0;

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

length++;

return length;

}

// поиск символа в строке

int indexOf(char ch, char\* string) {

for (int i = 0; i < getLength(string); i++)

if (string[i] == ch) return i;

return -1;

}

// индекс строки в массиве

int indexOf(char\* str, char\*\* strArr, int size) {

for (int i = 0; i < size; i++)

if (!strcmp(str, strArr[i]))

return i;

return -1;

}

// удаление гласных

char\* deleteVowels(char\* str, char\* vw = vowels) {

//char\* temp = (char\*)malloc(sizeof(char) \* maxLength);

char\* temp = new char[getLength(str)];

int cx = 0;

for (int i = 0; i < getLength(str); i++)

if (indexOf(str[i], vw) < 0)

temp[cx++] = str[i];

temp[cx] = '\0';

return temp;

}

// ввод массива символов

char\* getCharArray(int length, const char delim = '\n') {

char\* temp = new char[length];

int i = 0;

for (; i < length; i++) {

temp[i] = cin.get();

if (temp[i] == delim) break;

}

temp[i] = '\0';

return temp;

}

// вычленение слова из строки

char\* getWord(char\* str, char\*& newStr, char\*\*& delims, int& delimsSize, char\* delim = " ") {

char\* word = new char[strlen(str)];

delims = (char\*\*)realloc(delims, sizeof(char\*) \* ++delimsSize);

//delims[delimsSize - 1] = (char\*)realloc(delims[delimsSize - 1], sizeof(char) \* 1);

delims[delimsSize - 1] = new char[1];

int cx = 0;

int i = 0;

while (str[i] != '\0' && indexOf(str[i], delim) >= 0) {

delims[delimsSize - 1] = (char\*)realloc(delims[delimsSize - 1], sizeof(char) \* (i + 1));

delims[delimsSize - 1][i] = str[i];

i++;

}

delims[delimsSize - 1][i] = '\0';

while (str[i] != '\0' && indexOf(str[i], delim) < 0) {

word[cx++] = str[i++];

}

newStr = new char[strlen(str) - cx];

word[cx] = '\0';

cx = 0;

while (str[i] != '\0') {

newStr[cx++] = str[i++];

}

newStr[cx] = '\0';

return word;

}

// проверка число ли это

int isNumber(char\* str) {

if (str == NULL || strlen(str) == 0) return 0;

for (int i = 0; i < strlen(str); i++)

if (str[i] - '0' < 0 || str[i] - '0' > 9) return 0;

return 1;

}

// ввод числа с проверкой на диапазон

int input(int min, int max) {

char temp[256];

char\* cArray = NULL;

while (!isNumber(cArray) || stoi(cArray) > max || stoi(cArray) < min) {

cout << "Введите целое число от " << min << " до " << max << ": ";

cin.getline(temp, 256);

cArray = temp;

}

return stoi(cArray);

}

// добавление элемента в массив

void append(int\*& arr, int& size, int number) {

arr = (int\*)realloc(arr, sizeof(int) \* ++size);

arr[size - 1] = number;

}

// печать массива целых чисел

void print(int\* arr, int size, char\* delim = ", ") {

if (arr == NULL) {

cout << "Массив пуст...";

return;

}

cout << "[";

for (int i = 0; i < size - 1 && size > 1; i++)

cout << arr[i] << delim;

cout << arr[size - 1] << "]";

}

// печать массива строк

void print(char\*\* arr, int size, char\* delim = ", ") {

if (arr == NULL) {

cout << "Массив пуст...";

return;

}

cout << "[";

for (int i = 0; i < size - 1 && size > 1; i++)

cout << "\"" << arr[i] << "\"" << delim;

cout << "\"" << arr[size - 1] << "\"]";

}

// пересборка строки с массивом удаленных слов

char\* rebuildString(char\*\* words, int wordsSize, char\*\* delims, int delimsSize) {

char\* result = new char[0];

int cx = 0;

int length = 0;

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

length = length + strlen(delims[i]) + strlen(words[i]);

result = (char\*)realloc(result, sizeof(char) \* length);

for (int j = 0; j < strlen(delims[i]); j++)

result[cx++] = delims[i][j];

for (int j = 0; j < strlen(words[i]); j++)

result[cx++] = words[i][j];

}

result[length] = '\0';

return result;

}

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

cout << "Strings (lab. 3) Akzhigitov Radmir IKBO-07-17 (31.03.18)\n";

while (true) {

cout << "1) Удалить из строки гласные буквы\n2) Удалить повторяющиеся слова и выделить массив целый чисел\n3) Выход\n";

int choice = input(1, 3);

switch (choice) {

case 1 : {

cout << "Введите строку: ";

char\* str = getCharArray(maxLength);

char\* string = deleteVowels(str);

cout << string << endl;

delete[] string;

delete[] str;

system("pause");

system("cls");

break;

}

case 2 : {

cout << "Введите строку: ";

char\* str = getCharArray(maxLength);

char\* newStr = 0;

char\* word = 0;

int\* ints = 0;

int intsLength = 0;

char\*\* dictionaryKeys = 0; int dictionaryLength = 0;

int\* dictionaryValues = 0;

char\*\* delims = 0; int delimsSize = 0;

char\*\* words = new char\*[0];

int cx = 0;

int wordsCounter = 0;

char\* newString = new char[strlen(str)];

cout << "Введенная строка: " << str << endl;

while ((word = getWord(str, newStr, delims, delimsSize, " ,")) && strlen(str) != 0) {

str = newStr;

words = (char\*\*)realloc(words, sizeof(char\*) \* ++wordsCounter);

words[wordsCounter - 1] = new char[strlen(word)];

words[wordsCounter - 1] = word;

if (isNumber(word))

append(ints, intsLength, stoi(word));

cout << "Слово: " << word << " индекс в массиве слов: " << indexOf(word, dictionaryKeys, cx) << endl;

if (indexOf(word, dictionaryKeys, cx) >= 0) {

if (dictionaryValues[indexOf(word, dictionaryKeys, cx)] >= 0) {

words[wordsCounter - 1] = new char[1];

words[wordsCounter - 1] = "";

}

dictionaryValues[indexOf(word, dictionaryKeys, cx)]++;

}

else {

dictionaryValues = (int\*)realloc(dictionaryValues, sizeof(int) \* ++cx);

dictionaryValues[cx - 1] = 0;

dictionaryKeys = (char\*\*)realloc(dictionaryKeys, sizeof(char\*) \* cx);

//dictionaryKeys[cx - 1] = (char\*)malloc(dictionaryKeys[cx - 1], sizeof(char) \* strlen(word));

dictionaryKeys[cx - 1] = new char[strlen(word)];

dictionaryKeys[cx - 1] = word;

}

}

cout << "Массив чисел: ";

print(ints, intsLength);

cout << endl << "Ключи словаря: ";

print(dictionaryKeys, cx);

cout << endl << "Значения словаря: ";

print(dictionaryValues, cx);

cout << endl << "Разделители между словами: ";

print(delims, delimsSize);

cout << endl << "Слова: ";

print(words, wordsCounter);

cout << endl << "Удалены повторения: " << rebuildString(words, wordsCounter, delims, delimsSize) << endl;

delete[] ints;

delete[] dictionaryKeys; delete[] dictionaryValues;

system("pause");

system("cls");

break;

}

case 3 : {

break;

}

}

if (choice == 3) break;

}

system("pause");

return 0;

}

# Вывод

Строки – важный тип данных, хранится в памяти, как обычный массива символов, но в конце обязательно идет спецсимвол конца строки: «\0». Поэтому не обязательно указывать длину строки, программа считает строку до этого символа. Множество методов можно найти в модуле <string>. Ввод строки можно осуществлять с помощью поток ввода cin >>, cin.getline(char\* arr[size], size), getline(cin, std::string str).