Часть 1:  
  
#include <bits.h>

#include <iostream>

#include <bitset>

#include <sstream>

#include <algorithm>

using namespace std;

int main()

{

setlocale(LC\_ALL, "Russian");///<-- Включила русский язык

cout << "Enter the Name: ";/// <-- Прошу ввести имя

string Word;/// <--объявила строковую переменную Word, в ней будет храниться имя

getline(cin, Word); /// <-- Получила имя

int lenth\_Word = Word.length();///<-- Вычислила его длину

/\* Хочу перевести строку в символьный массив.

Это необходимо, тк мы будем работать в первым и последним символами имени \*/

char\* chars = new char[lenth\_Word];/// <-- Создала динамический символьный массив chars, длина которого равна длине имени

Word.copy(chars, lenth\_Word);/// <-- Скопировала строку имени в массив по-символьно

/\* Дальше нужно получить по-битовое представление двух символов \*/

bitset<8> char1; bitset<8> char2;

char1 = bitset<8>(chars[0]); char2 = bitset<8>(chars[lenth\_Word-1]);

/\* Тоесть я объявила, что в char1 и char2 буду помещать по-битовые представления,

а затем дала переменным char1 и char 2 значения этих самых по-битовых представлений \*/

string Char1 = char1.to\_string<char, char\_traits<char>, allocator<char> >();

string Char2 = char2.to\_string<char, char\_traits<char>, allocator<char> >();

/\* Объявила строки Char1 и Char2, поместила в них по-битовые представления символов \*/

/\* В итоге у нас две строки: Char1 и Char2, в которых хранятся биты первого и последнего символа имени

в строковом виде соответственно \*/

/\* Теперь я эти строки хочу перевести в символьные массивы

Возможно, можно было как-то обойтись вообще без строк Char1 и Char2, сразу отправив char1 и char2 в массивы,

но мои познания ограничены \*/

char\* Bit1 = new char[8]; char\* Bit2 = new char[8];/// <-- Объявила динамические символьные массивы размером 8 эл-в

Char1.copy(Bit1, 8); Char2.copy(Bit2, 8); /// <-- Поместила в них строки с битами

/\* Дальше разбираемся с паритетными битами.

Т.к. биты принимают значения 1 или 0 (true или false), взяла булевый тип.\*/

bool Flag1 = 0; bool Par\_Plus\_Bit1; bool Par\_Minus\_Bit1; /// <--Создаю булевую переменную для положительного парит.бита и для отрицательного

/\* Выше я так же создала булевую переменную Flag1, она нужна для выполнения условия о чётности/нечётности

Работает это так:изначально Flag1= 0, если в массиве из битов встретилась 1, то Flag1 =1 (команда Flag1++ увеличивает значение на один),

если 1 встретилась снова, то Flag= 0 (тк если к булевой переменной, равной 1 применить ++, то переменная снова примет значение 0)

Таким образом, когда кол-во 1 нечётно, Flag1= 1, а когда кол-во чётно, то Flag1= 0 \*/

for (int i = 0; i < 8; i++)

{

if (Bit1[i] == 1) { Flag1++; }/// <-- Встретили 1, увеличили Flag1

}

/\*

Дальше нужно выполнить условие

Если Flag1 в итоге равен 1, значит единица встретила нечётное кол-во раз,

соответственно чётный паритетный бит станет true

Если Flag1 в итоге равен 0, значит единица встретила чётное кол-во раз,

соответственно чётный паритетный бит станет false

if (Flag1 == 1) { Par\_Plus\_Bit1 = 1; Par\_Minus\_Bit1 = 0; }

else { Par\_Plus\_Bit1 = 0; Par\_Minus\_Bit1 = 1; }

/\* Со вторым символом проделываем то же самое \*/

bool Flag2 = 0; bool Par\_Plus\_Bit2; bool Par\_Minus\_Bit2;

for (int i = 0; i < 8; i++)

{

if (Bit2[i] == 1) { Flag2++; }

}

if (Flag2 == 1) { Par\_Plus\_Bit2 = 1; Par\_Minus\_Bit2 = 0; }

else { Par\_Plus\_Bit2 = 0; Par\_Minus\_Bit2 = 1; }

cout << endl; system("pause"); system("cls");/// <-- Очищаю экран (нужно нажать на любую кнопку)

/\* Осталось вывести на экран.\*/

cout << " Буква " << "Битовая строка " << "Чётный паритетный бит " << "нечётный паритетный бит"<<endl<<endl;

cout << " " << chars[0] << " " << Char1 << " " << Par\_Plus\_Bit1 << " " << Par\_Minus\_Bit1 << endl;

cout << " " << chars[lenth\_Word-1] << " " << Char2 << " " << Par\_Plus\_Bit2 << " " << Par\_Minus\_Bit2 << endl;

}