МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное автономное образовательное учреждение высшего образования
"Омский государственный технический университет"

Кафедра «Автоматизированные системы обработки информации и управления»

ПРАКТИЧЕСКАЯ РАБОТА №1

по дисциплине

«Измерительные средства аналитики программных систем и технологий»

по теме

«Метрики Холстеда.»

Выполнил: Шмидт А.В.
гр. ИВТ - 244
Проверил: Зубарев А.А.
к.т.н., доцент

Омск 2025

ЗАДАНИЕ

Рассчитать характеристики реализации алгоритма, выполненной на одном знакомом языке программирования. Предложить смысловую трактовку как используемых данных, так и описываемых характеристик.

ПРИМЕР ОПРЕДЕЛЕНИЯ МЕТРИК ПО МОДЕЛИ ХОЛСТЕДА

Задачей данной программы, выполненной на языке программирования C++, является анализ строки длиной до 100 символов, вводимой пользователем с клавиатуры. Программа вычисляет длину строки, количество гласных букв (a, e, i, o, u, y — без учёта регистра) и определяет самую часто встречающуюся букву (игнорируя пробелы, цифры и знаки препинания). Результаты выводятся на экран. После завершения анализа программа ожидает нажатия клавиши Enter, чтобы консольное окно не закрывалось сразу после выполнения. Ниже представлен листинг программного модуля.

```cpp
#include <iostream>
#include <string>
#include <cctype>

using namespace std;

int main() {
    string input;
    cout << "Enter a string (up to 100 characters): ";
    getline(cin, input);

    if (!input.empty() && input.back() == '\n') {
        input.pop_back();
    }

    int length = input.length();
    int vowels = 0;
    int freq[26] = {0};
    char most_common = '\0';
    int max_count = 0;

    string vowel_chars = "aeiouyAEIOUY";

    for (char c : input) {
        if (isalpha(c)) {
            if (vowel_chars.find(c) != string::npos) {
                vowels++;
            }

            char lower = tolower(c);
            int index = lower - 'a';
            freq[index]++;
            if (freq[index] > max_count) {
                max_count = freq[index];
                most_common = lower;
            }
        }
```

```
    }

    cout << "String length: " << length << '\n';
    cout << "Number of vowels: " << vowels << '\n';

    if (max_count > 0) {
        cout << "Most frequent letter: '" << most_common
             << "' (appears " << max_count << " time";
        if (max_count > 1) cout << "s";
        cout << ")" << '\n';
    } else {
        cout << "Most frequent letter: (no letters)" << '\n';
    }

    cout << "\nPress Enter to exit...";
    cin.get();

    return 0;
}
```

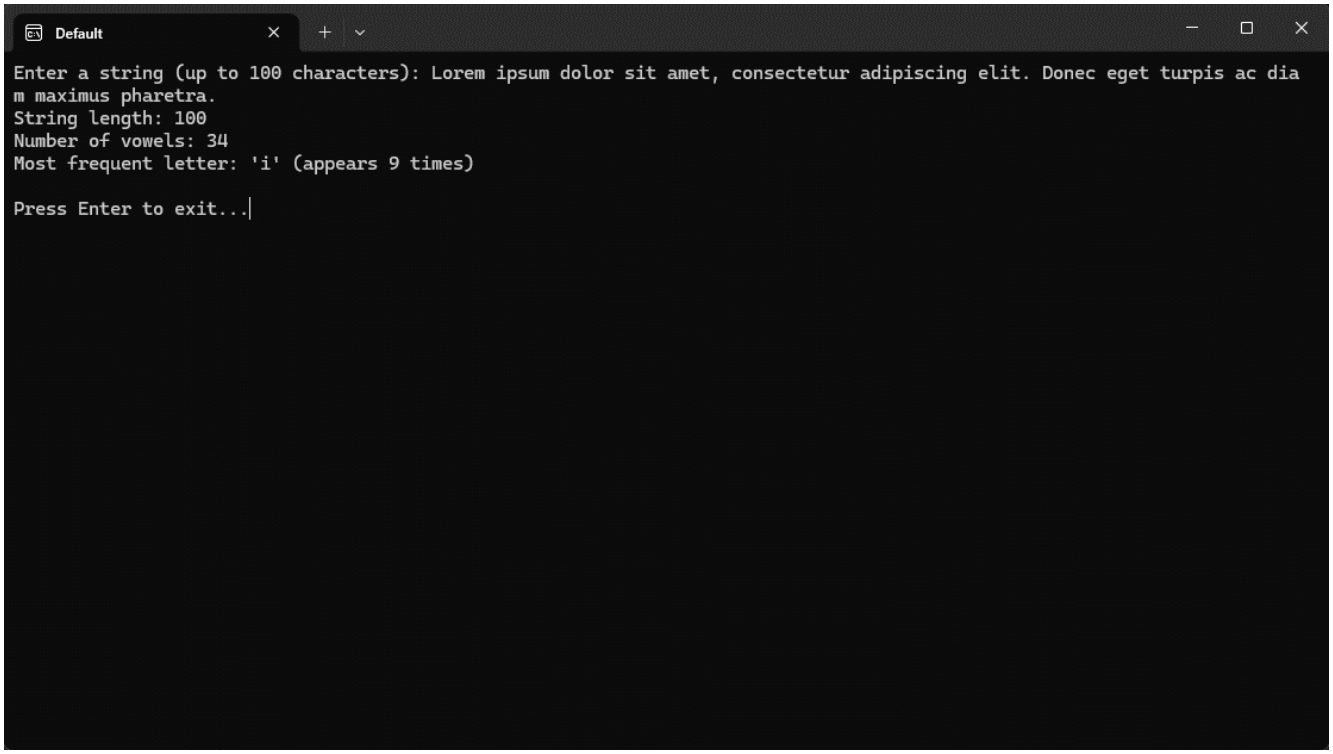

Рисунок 1 – Результат работы программы.

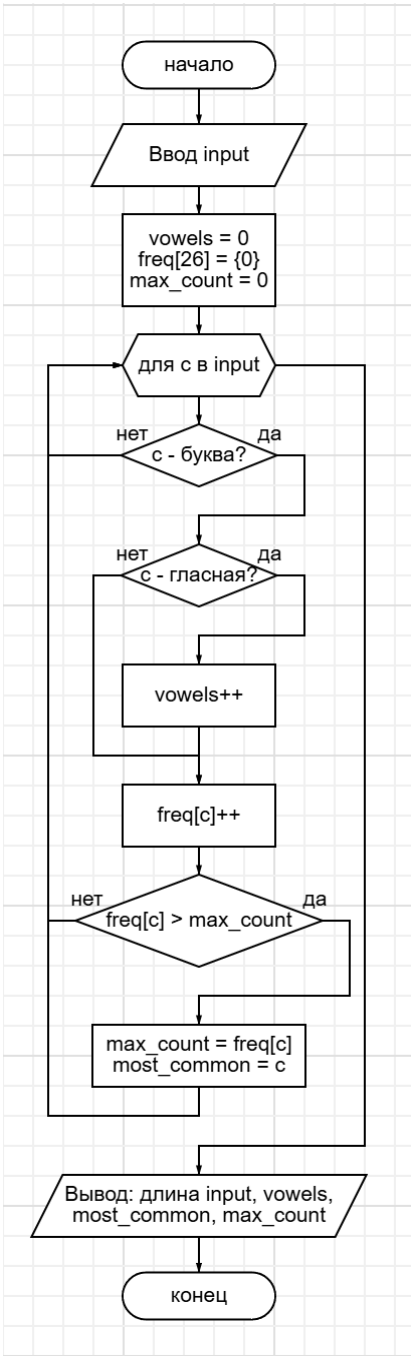

Рисунок 2 – Схема алгоритма программы.

Результаты подсчёта числа типов операторов и операндов для рассматриваемой программы и их общего количества представлены в таблицах 1 и 2.

Таблица 1 – Подсчёт числа типов операторов.

Оператор	Уникальный n1	N2
;	1	26
,	2	1
{}	3	9
()	4	20
=	5	13
++	6	2
-	7	1
[]	8	4
for	9	1
int	10	6
<	11	3
>	12	3
if	13	6
else	14	1
return	15	1
include	16	3
<<	17	18
&&	18	1
!=	19	1
.	20	6
::	21	1
!	22	1
string	23	3

Продолжение таблицы 1 – Подсчёт числа типов операторов.

Оператор	Уникальный n1	N1
char	24	3
cout	25	8
cin	26	1
getline	27	1
	27	144
	n1	N1

Таблица 2 – Подсчёт числа типов операндов.

Операнд	Уникальный n2	N2
input	1	7
length	2	3
vowels	3	3
freq	4	4
most_common	5	3
max_count	6	6
vowel_chars	7	2
c	8	4
lower	9	3
index	10	4
	10	39
	n2	N2

Результаты подсчёта числа операторов и операндов приведены в таблице 3.

Таблица 3 – Результат подсчёта числа операторов и операндов.

Название данных	Частота данных
Число уникальных операторов (n1)	27
Число уникальных операндов (n2)	10
Общее число операторов (N1)	144
Общее число операндов (N2)	39
Словарь программы (n)	37
Количество входных, выходных переменных (n2*)	10

Учитывая данные характеристики, Холстедом были введены следующие оценки:

словарь программы: n = n1 + n2 = 27 + 10 = 37;

длина программы: N = N1 + N2 = 144 + 39 = 183;

объём программы: V = Nlog2(n) = 183 * log2(37) = 954 бит;

теоретическая длина программы: NT = n1log2(n1) + n2log2(n2) = 27*log2(27) + 10*log2(10) = 161,10;

экспериментальная длина программы: Nэ = N1 + N2 = 144 + 39 = 183;

потенциальный объем программы: V* = (n2* + 2)*log2(n2* + 2) = (10 + 2)*log2(10 + 2) = 43,02 бит;

уровень качества программирования: L = V* / V = 43,02 / 954 = 0,05;

сложность программы: S = 1 / L = 1 / 0,05 = 20;

уровень программы: L^ = (2n2) / (n1N2) = (2 * 10) / (27 * 39) = 0,019;

интеллект программы: I = L^V = 0,019 * 954 = 18,126;

работа по программированию: E = VS = 954 * 20 = 19080 [количество элементарных операций по осмыслению];

время кодирования: T = E / Str = 19080 / 10 = 1908, где Str = 10 — число Страуда.

ожидаемое время кодирования: T = n1N2(n1log2n1 + n2log2n2)log2n / 2n2Str = 27*39(27*log2(27) + 10*log2(10)) / 2*10*18 = 472,68;

уровень языка программирования: λ = LV* = 0,05 * 43,02 = 2,15;

ожидаемый уровень ошибок: B = V / 3000 = 954 / 3000 = 0,318, где 3000 — среднее число элементарных различений между возможными ошибками при программировании;

оптимальная модульность: M = n2* / 6 = 10 / 6 = 1,67;

прогноз числа ошибок: B = Nlog2(n / 3000) = 183*log2(37 / 3000) = -1160,2;

Для наглядности результаты, описанные выше, были внесены в таблицу 4.

Таблица 4 – Результаты подсчёта по метрикам Холстеда.

Название метрики	Результат
Словарь программы n	37
Длина программы N	183
Объем программы V	954
Теоретическая длина программы Nт	161,10
Экспериментальная длина программы Nэ	183
Потенциальный объем программы V*	23,02
Уровень качества программирования L	0,05
Сложность программы S	20
Уровень программы L^	0,019
Интеллект программы I	18,126
Работа по программированию E	19080
Время кодирования T	1908
Ожидаемое время кодирования T	472,68
Уровень языка программирования □	2,15
Ожидаемый уровень ошибок B	0,318
Оптимальная модульность M	1,67
Прогноз числа ошибок B	-1160,2

ВЫВОД

В рамках практической работы рассчитаны метрики Холстеда для программы на C++, анализирующей строку длиной до 100 символов, определяющей её длину, количество гласных и наиболее частую букву. Словарь программы (n = 37), длина (N = 183), объём (V = 954 бит), теоретическая длина (N_T = 161.10), потенциальный объём (V* = 43.02 бит), уровень качества (L = 0.05), сложность (S = 20), уровень программы (L^ = 0.019), интеллект (I = 18.126), работа (E = 19080), время кодирования (T = 1908) и ожидаемое время (T = 472.68) отражают умеренную сложность и информационную насыщенность кода. Ожидаемый уровень ошибок (B = 0.318) и отрицательный прогноз ошибок (B = -1160.2) указывают на ограничения модели Холстеда для данной задачи. Программа демонстрирует сбалансированную реализацию, подходящую для учебных целей, но требует осторожной интерпретации метрик ошибок. Метрики подтверждают применимость модели Холстеда для оценки качества кода и трудозатрат, а для улучшения программы рекомендуется оптимизация структуры и повышение уровня абстракции.