Министерство науки и высшего образования РФ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«Омский государственный технический университет»

Кафедра «Автоматизированные системы обработки информации и управления»

# ПРАКТИЧЕСКАЯ РАБОТА №2

по дисциплине

“Измерительные средства аналитики программных систем и технологий”

по теме:

“**Метрики Холстеда**”

Выполнил:
Моисеенко Д.В.
Студентка группы ИВТ- 234
Проверил:
Зубарев А.А., к.т.н., доцент

Омск 2024

## 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 поиск минимального, максимального и среднего арифметического значений массива, размер и элементы которого с клавиатуры указывает пользователь. Ниже представлен листинг программного модуля.

```c
#include <stdio.h>

int main()
{
    int N;
    printf("Input array size: ");
    scanf("%d", &N);

    int arr[N];

    for (int i = 0; i < N; i++)
    {
        printf("Input [%d] element of array: ", i+1);
        scanf("%d", &arr[i]);
    }

    int min_arr = 999;
    int max_arr = -999;
    float avg_arr = 0;

    int sum = 0.0;
    for (int i = 0; i < N; i++)
    {
        sum += arr[i];

        if (arr[i] < min_arr)
        {
            min_arr = arr[i];
        }

        if (arr[i] > max_arr)
        {
            max_arr = arr[i];
        }
        else
        {
            continue;
        }
    }

    avg_arr = sum / N;

    printf("Min element: %d\n", min_arr);
    printf("Max element: %d\n", max_arr);
    printf("Average: %.2f\n", avg_arr);

    return 0;
}
```

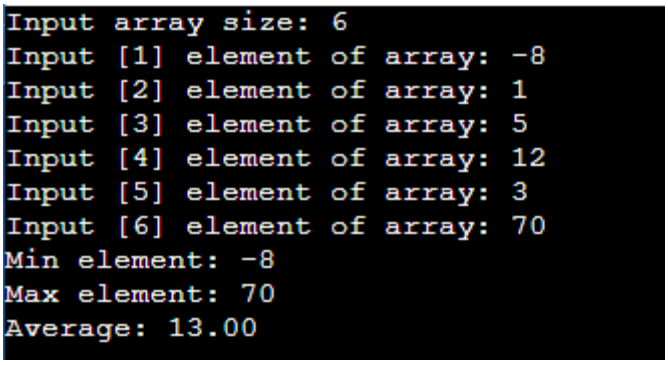


Рисунок 1 — Работа программы

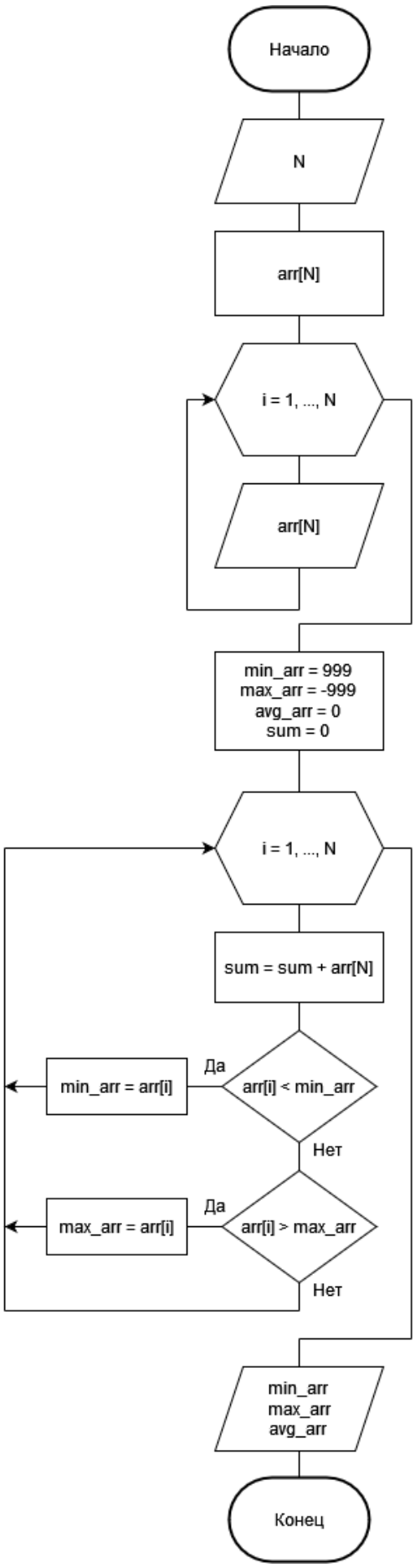


Рисунок 2 — Схема алгоритма рассматриваемой программы

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

Таблица 1 — Операторы программы

| Оператор | Уникальный $n_1$ | $N_1$ |
|---|---|---|
| ; | 1 | 23 |
| , | 2 | 6 |
| {} | 3 | 6 |
| () | 4 | 12 |
| = | 5 | 9 |
| += | 6 | 1 |
| ++ | 7 | 2 |
| + | 8 | 1 |
| - | 9 | 1 |
| / | 10 | 1 |
| [] | 11 | 6 |
| & | 12 | 2 |
| for (int i = 0; i < N; i++) | 13 | 2 |
| int | 14 | 7 |
| float | 15 | 1 |
| < | 16 | 3 |
| > | 17 | 1 |
| if (arr[i] < min_arr) | 18 | 1 |
| if (arr[i] > max_arr) | 19 | 1 |
| else | 20 | 1 |
| continue | 21 | 1 |
| printf | 22 | 5 |
| scanf | 23 | 2 |
| return | 24 | 1 |
| include | 25 | 1 |
| | 25 | 97 |
| | $n_1$ | $N_1$ |

Таблица 2 — Операнды программы

| Операнд | Уникальный $n_2$ | $N_2$ |
|---|---|---|
| N | 1 | 5 |
| arr | 2 | 5 |
| i | 3 | 13 |
| min_arr | 4 | 4 |
| max_arr | 5 | 4 |
| avg_arr | 6 | 3 |
| sum | 7 | 3 |
| | 7 | 37 |
| | $n_2$ | $N_2$ |

Результаты подсчёта числа операторов и операндов приведены в таблице 3.

Таблица 3 — Общее количество операторов и операндов

| Название данных | Частота данных |
|---|---|
| Число уникальных операторов ($n_1$) | 25 |
| Число уникальных операндов ($n_2$) | 7 |
| Общее число операторов ($N_1$) | 97 |
| Общее число операндов ($N_2$) | 37 |
| Словарь программы (n) | 32 |
| Количество входных, выходных переменных ($n_2$*) | 7 |

Учитывая данные характеристики, Холстедом были введены следующие оценки:

**словарь программы**: $n = n_1 + n_2 = 25 + 7 = 32$;

**длина программы**: $N = N_1 + N_2 = 97 + 37 = 134$;

**объём программы**: $V = N\log_2(n) = 134 * \log_2(32) = 670$ бит;

**теоретическая длина программы**: $N_T = n_1\log_2(n1) + {}_{n2\log2}(n_2) = 25*\log_2(25) + 7*\log_2(7) = 135{,}75$;

**экспериментальная длина программы**: Nэ = $N_1 + N_2 = 97 + 37 = 134$;

**потенциальный объем программы**: $V^* = (n_2^* + 2)\log_2(n_2^* + 2) = (7 + 2)\log_2(7 + 2) = 28{,}53$ бит;

**уровень качества программирования**: $L = V^* / V = 28{,}53 / 670 = 0{,}04$;

**сложность программы**: $S = 1 / L = 1 / 0{,}04 = 25$;

**уровень программы**: $L\hat{} = (2n_2) / (n_1N_2) = (2 * 7) / (25 * 37) = 0{,}015$;

**интеллект программы**: $I = L\hat{}V = 0{,}015 * 670 = 10{,}05$;

**работа по программированию**: $E = VS = 670 * 25 = 16750$ [количество элементарных операций по осмыслению];

**время кодирования**: $T = E / S_{tr} = 16750 / 10 = 1675$, где $S_{tr} = 10$ — число Страуда.

**ожидаемое время кодирования**: $T = n_1N_2(n_1\log_2 n_1 + n_2\log_2 n_2)\log_2 n / 2n_2S_{tr} = 25*37(25*\log_2(25) + 7*\log_2(7)) / 2*7*18 = 498{,}3$;

**уровень языка программирования**: $\lambda = LV^* = 0{,}04 * 28{,}53 = 1{,}14$;

**ожидаемый уровень ошибок**: $B = V / 3000 = 670 / 3000 = 0{,}223$, где 3000 — среднее число элементарных различений между возможными ошибками при программировании;

**оптимальная модульность**: $M = n_2^* / 6 = 7 / 6 = 1{,}16$;

**прогноз числа ошибок**: $B = N\log_2(n / 3000) = 134*\log2(32 / 3000) = -877{,}8$;

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

Таблица 4 — Рассчитанные значения метрик Холстеда

| Название метрики | Результат |
|---|---|
| Словарь программы n | 32 |
| Длина программы N | 134 |
| Объем программы V | 670 |
| Теоретическая длина программы $N_т$ | 135,75 |
| Экспериментальная длина программы $N_э$ | 134 |

Окончание таблицы 4

| Название метрики | Результат |
|---|---|
| Потенциальный объем программы V* | 28,53 |
| Уровень качества программирования L | 0,04 |
| Сложность программы S | 25 |
| Уровень программы L^ | 0,015 |
| Интеллект программы I | 10,05 |
| Работа по программированию E | 16750 |
| Время кодирования T | 1675 |
| Ожидаемое время кодирования T | 498,3 |
| Уровень языка программирования $\lambda$ | 1,14 |
| Ожидаемый уровень ошибок B | 0,223 |
| Оптимальная модульность M | 1,16 |
| Прогноз числа ошибок B | -877,8 |

# ВЫВОД

1. Метрики теоретической длины и экспериментальной длины программы можно использовать для выявления недостатков программирования. Для корректно написанного кода относительная разность теоретической и экспериментальной длины программы (по Холстеду) лежит в пределах 10–12 %.

В данной программе теоретическое значение длины программы $N_T$ = 135,75 и экспериментальное значение длины $N_Э$ = 134 отличаются на 1.29%, поэтому по Холстеду можно сказать, что в программе нет стилистических ошибок (несовершенств).

2. Размер объёма программы для одной функции рекомендуется от 20 до 1000, а для одного файла – от 100 до 8000. Если размер этой метрики превышает верхнюю границу диапазона, то рекомендуется разбить исследуемый элемент на несколько составляющих.

В данной программе объем составляет 670 и укладывается в рекомендуемый диапазон.

3. Потенциальный объём зависит от количества входных и выходных параметров $n_2$* = 7 и не зависит от языка реализации. V* = 28,53 соответствует минимально компактному тексту.

При переводе алгоритма с одного языка программирования на другой объем программы V будет изменяться в зависимости от рассматриваемых языков, а V* будет всегда оставаться неизменным.

4. Уровень качества программирования – это численное значение, показывающее краткость программы. Чем ближе значение L к единице, тем более совершенна программа. Для идеальной программы L = 1, а для реальной L < 1.

Уровень рассматриваемой программы равен 0,04, что значительно меньше, чем 1. Это свидетельствует о том, что программа является некомпактной.

5. Уровень программы по параметрам реальной программы – это уровень качества программирования, основанный лишь на параметрах реальной программы без учета теоретических параметров. Эта метрика применяется, когда нужно определить уровень программирования, не прибегая к оценке её теоретического объема.

Уровень программы по параметрам реальной программы составляет 0,015. В соотношении с L этот уровень примерно в два с половиной раза выше.

6. Интеллект программы – это оценка необходимых интеллектуальных усилий при разработке программы, и метрика характеризует число требуемых элементарных решений при написании программы (оценка мыслительных затрат).

В рассмотренной программе I = 10,5. Это свидетельствует о том, что для создания программы не нужно прикладывать больших умственных затрат.

7. Работа по программированию составляет 16750. Эта характеристика показывает количество элементарных операций, которые совершает человеческий мозг при реализации программы.

8. Время кодирования T и ожидаемое время кодирования T^ различаются в данном примере практически в два раза: T = 1675 = 3 ч, T^ = 498,3 = 2 ч. Это может зависеть от языка программирования. Когда Холстед разрабатывал данные метрики, языки программирования требовали более объемного и сложного алгоритма в реализации, дополнительных переменных и операций.

9. Уровень языка программирования $\lambda$ = 1,14. Так как произведение L на V* остается неизменным для любого языка, следовательно, λ уменьшается пропорционально увеличению объема программы. Объем программы, реализованной на языке низкого уровня, будет больше, чем на языке высокого уровня, а уровень языка программирования будет меньше.

10. Ожидаемый уровень ошибок B составляет 0,223. Данная метрика показывает возможные ошибки, оставшиеся в реализации программы после некоторой идентифицирующей фазы. Поскольку данный метод предсказывает

лишь число первоначальных ошибок, то он не сходится при их исключении и не дает «доказательства правильности» в математическом смысле, но обеспечивает высокую степень доверия в прикладном отношении. В примере вероятно не передана и одна ошибка.

Время, требуемое на реализацию программы для ЭВМ, зависит от числа элементарных мысленных различений E, необходимых программисту. Следовательно, число моментов, в которые можно сделать ошибочное различение, также определяется значением E.

11. Оптимальная модульность M = 1,16, и она показывает количество модулей, на которые желательно разбить программу. Это принцип, согласно которому логически связанные между собой подпрограммы, переменные и т. д. группируются в отдельные модули. Разбиение на модули упрощает понимание и снижает стоимость разработки. Модульность снижает работу по программированию. Пример не превышает объём одного модуля, но при добавлении малейшего функционала её было бы необходимо разбить на несколько файлов исходного кода.

12. Прогноз числа ошибок составляет -877,8. Уровень ошибок зависит от сложности программы. Из-за того, что в логарифм подставляется не просто словарь программы, а словарь программы, деленный на критический уровень работы (по ошибкам) при составлении программы, то в результате значение прогноза числа ошибок получается отрицательным. При прогнозе, равном нулю, предполагается, что передана одна ошибка. При полученном отрицательном значении прогноза передача ошибки не предполагается.

Метрики Холстеда, как правило, вычисляются уже на основе готового исходного кода и применимы для получения апостериорных оценок, т. е. на основании опыта. Они могут быть использованы для того, чтобы предотвратить чрезмерное усложнение отдельных структурных элементов программного проекта и роста связанных с этим рисков. С помощью этих метрик могут быть получены основные характеристики качества программ.