Федеральное агентство связи
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Сибирский государственный университет телекоммуникаций и информатики»

Лабораторная работа №12

Выполнил:
студенты 4 курса
группы ИП-216
Андрущенко Ф.А.
Литвинов А. Е.
Русецкий А. С.

Проверил:
преподаватель кафедры ПМиК
Агалаков Антон Александрович

Новосибирск, 2025 г.

Задание

Задание

1. Написать подпрограммы на двух языках программирования для решения следующих задач:

ЗАДАЧА
1. Отыскать минимальный элемент одномерного массива целых, его значение и значение его индекса.
2. Сортировка одномерного массива в порядке возрастания методом пузырька.
3. Бинарный поиск элемента в упорядоченном одномерном массиве.
4. Отыскать минимальный элемент двумерного массива целых, его значение и значение его индексов.
5. Осуществить перестановку значений элементов одномерного массива в обратном порядке.
6. Осуществлять циклический сдвиг элементов одномерного массива на заданное число позиций влево.
7. Заменить все вхождения целочисленного значения в целочисленный массив.

2. Для каждой подпрограммы вычислить следующие метрические характеристики:

- η^*_2 – число единых по смыслу входных и выходных параметров, представленных в сжатой без избыточной форме;
- η_1 - число отдельных операторов;
- η_2 - число отдельных операндов;
- η - длина словаря реализации;
- N_1 - общее число вхождений всех операторов в реализацию;
- N_2 - общее число вхождений всех операндов в реализацию;
- N - длина реализации;
- $N^\wedge$ - предсказанная длина реализации по соотношению Холстеда;
- V^* - потенциальный объем реализации:

$$V^* = (2+\eta_2^*) * \log_2(2+\eta_2^*).$$

- V - объем реализации:

$$V = N * \log_2 \eta.$$

- L - уровень программы через потенциальный объем:

$$L = V^* / V.$$

- $L^\wedge$ - уровень программы по реализации:

$$L^\wedge = (2/\eta_1) * (\eta_2 / N_2).$$

- I - интеллектуальное содержание программы:

$$I = (2/\eta_1) * (\eta_2 / N_2) * (N_1 + N_2) * \log_2(\eta_1 + \eta_2).$$

- $T^\wedge_1$ - прогнозируемое время написания программы, выраженное через потенциальный объем:

$$\hat{T} = \frac{V^2}{S * V^*}.$$

- $T^\wedge_2$ - прогнозируемое время написания программы, выраженное через длину реализации, найденную по Холстеду (т.е. в предположении, что программа совершенна):

$$\hat{T} = \frac{\eta_1 \times N_2 \times (\eta_1 \log_2 \eta_1 + \eta_2 \log_2 \eta_2) \times \log_2 \eta}{2 \times S \times \eta_2}.$$

- $T^\wedge_3$ - прогнозируемое время написания программы, выраженное через метрические характеристики реализации:

$$\hat{T} = \frac{\eta_1 \times N_2 \times N \times \log_2 \eta}{2 \times S \times \eta_2}.$$

3. По всем реализациям алгоритмов определить средние значения уровней языков программирования λ:

$$\lambda_1 = \hat{L}^2 \times V,$$

$$\lambda_2 = \frac{V^{*2}}{V}.$$

Содержание отчета

Для каждой подпрограммы отчет должен содержать:

- текст подпрограммы,
- метрические характеристики реализации и алгоритма, оформленные в виде следующей таблицы:

η^*_2	η_1	η_2	η	N_1	N_2	N	$N^\wedge$	V^*	V	L	$L^\wedge$	I	$T^\wedge$	$T^\wedge$	$T^\wedge$	λ_1	λ_2

- анализ полученных результатов.

Контрольные вопросы

Знать и уметь объяснить смысл формул, которые использовались для вычисления метрических характеристик при выполнении лабораторной работы.

Литература

1. Холстед М.Х. Начала науки о программах. - М.: Финансы и статистика, 1981.
2. Кайгородцев Г.И. Введение в курс метрической теории и метрологии программ. - Новосибирск: НГТУ, 2009.

Исходный код программы

```
import tokenize
import io
import keyword
import builtins
import math
from typing import List, Tuple, Dict, Any
import inspect

S = 18


class HalsteadMetrics:
    def __init__(self):
        self.python_keywords = set(keyword.kwlist)
        self.builtin_names = set(dir(builtins))

    def extract_tokens(self, code: str) -> Tuple[List[str], List[str]]:
        """Извлекает операторы и операнды из кода функции"""
        operators = []
        operands = []

        try:
            f = io.BytesIO(code.encode("utf-8"))
            for tok in tokenize.tokenize(f.readline):
                if tok.type in (
```

```
                    tokenize.COMMENT,
                    tokenize.NL,
                    tokenize.NEWLINE,
                    tokenize.ENCODING,
                    tokenize.ENDMARKER,
                ):
                    continue

                token_str = tok.string

                if tok.type == tokenize.OP:
                    operators.append(token_str)
                elif tok.type == tokenize.NAME:
                    if token_str in self.python_keywords:
                        operators.append(token_str)
                    else:
                        operands.append(token_str)
                elif tok.type in (tokenize.NUMBER, tokenize.STRING):
                    operands.append(token_str)

        except tokenize.TokenError:
            pass

        return operators, operands

    def compute_metrics(self, code: str, eta2_star: int) -> Dict[str, Any]:
        """Вычисляет все метрики Холстеда для заданного кода"""
        operators, operands = self.extract_tokens(code)

        unique_operators = set(operators)
        unique_operands = set(operands)

        eta1 = len(unique_operators)
        eta2 = len(unique_operands)
        eta = eta1 + eta2

        N1 = len(operators)
        N2 = len(operands)
        N = N1 + N2

        N_hat = 0.0
        if eta1 > 0:
            N_hat += eta1 * math.log2(eta1)
        if eta2 > 0:
            N_hat += eta2 * math.log2(eta2)

        V_star = (2 + eta2_star) * math.log2(2 + eta2_star) if eta2_star > 0 else 0

        V = N * math.log2(eta) if eta > 0 else 0

        L = V_star / V if V > 0 else 0
        L_hat = (2 / eta1) * (eta2 / N2) if (eta1 > 0 and N2 > 0) else 0
```

```
        I = 0.0
        if eta1 > 0 and N1 > 0 and (eta1 + eta2) > 0:
            I = (2 / eta1) * (N2 / N1) * (eta1 + eta2) * math.log2(eta1 + eta2)

        T1 = V_star / S if S > 0 else 0

        T2 = 0.0
        if S > 0 and eta1 > 0 and eta2 > 0:
            T2 = (N_hat * (eta1 * math.log2(eta2) + eta2 * math.log2(eta1))) / (2 * S)

        T3 = 0.0
        if S > 0 and N1 > 0 and N2 > 0 and eta2 > 0:
            T3 = (N1 * N2 * math.log2(eta2)) / (2 * S)

        return {
            "eta2_star": eta2_star,
            "eta1": eta1,
            "eta2": eta2,
            "eta": eta,
            "N1": N1,
            "N2": N2,
            "N": N,
            "N_hat": N_hat,
            "V_star": V_star,
            "V": V,
            "L": L,
            "L_hat": L_hat,
            "I": I,
            "T1": T1,
            "T2": T2,
            "T3": T3,
        }


def task1_find_min(arr: List[int]) -> Tuple[int, int]:
    """1. Минимальный элемент одномерного массива и его индекс"""
    if not arr:
        raise ValueError("Array is empty")
    min_val = arr[0]
    min_idx = 0
    for i in range(1, len(arr)):
        if arr[i] < min_val:
            min_val = arr[i]
            min_idx = i
    return min_val, min_idx


def task2_bubble_sort(arr: List[int]) -> List[int]:
    """2. Сортировка пузырьком"""
    n = len(arr)
    arr = arr[:]
```

```
    for i in range(n):
        for j in range(0, n - i - 1):
            if arr[j] > arr[j + 1]:
                arr[j], arr[j + 1] = arr[j + 1], arr[j]
    return arr


def task3_binary_search(arr: List[int], target: int) -> int:
    """3. Бинарный поиск"""
    left, right = 0, len(arr) - 1
    while left <= right:
        mid = (left + right) // 2
        if arr[mid] == target:
            return mid
        elif arr[mid] < target:
            left = mid + 1
        else:
            right = mid - 1
    return -1


def task4_find_min_2d(matrix: List[List[int]]) -> Tuple[int, int, int]:
    """4. Минимальный элемент двумерного массива"""
    if not matrix or not matrix[0]:
        raise ValueError("Matrix is empty")
    min_val = matrix[0][0]
    min_i = min_j = 0
    for i in range(len(matrix)):
        for j in range(len(matrix[i])):
            if matrix[i][j] < min_val:
                min_val = matrix[i][j]
                min_i, min_j = i, j
    return min_val, min_i, min_j


def task5_reverse_array(arr: List[int]) -> List[int]:
    """5. Перестановка в обратном порядке"""
    return arr[::-1]


def task6_cyclic_shift_left(arr: List[int], k: int) -> List[int]:
    """6. Циклический сдвиг влево на k позиций"""
    if not arr:
        return arr
    k = k % len(arr)
    return arr[k:] + arr[:k]


def task7_replace_value(arr: List[int], old_val: int, new_val: int) -> List[int]:
    """7. Замена всех вхождений значения"""
    return [new_val if x == old_val else x for x in arr]
```

```python
TASKS = [
    (task1_find_min, 3),
    (task2_bubble_sort, 2),
    (task3_binary_search, 3),
    (task4_find_min_2d, 4),
    (task5_reverse_array, 2),
    (task6_cyclic_shift_left, 3),
    (task7_replace_value, 4),
]


def get_function_source(func) -> str:
    """Получает исходный код тела функции (без def и docstring)"""
    try:
        lines = inspect.getsourcelines(func)[0]
        code_lines = []
        after_def = False
        skip_docstring = False

        for line in lines:
            stripped = line.strip()

            if not after_def:
                if stripped.endswith(":"):
                    after_def = True
                continue

            if stripped.startswith('"""') or stripped.startswith("'''"):
                skip_docstring = not skip_docstring
                if stripped.endswith('"""') or stripped.endswith("'''"):
                    skip_docstring = False
                continue

            if skip_docstring:
                continue

            if stripped == "":
                continue

            code_lines.append(line)

        return "".join(code_lines)
    except:
        return inspect.getsource(func)


def main():
    metrics_calculator = HalsteadMetrics()

    all_L_hat = []
    all_V = []
```

```
    print("=" * 100)
    print("ЛАБОРАТОРНАЯ РАБОТА №12: МЕТРИКИ ХОЛСТЕДА")
    print("=" * 100)

    for i, (func, eta2_star) in enumerate(TASKS, 1):
        print(
            f"\n--- ЗАДАЧА {i}: {func.__doc__.strip() if func.__doc__ else 'Без
описания'} ---"
        )

        code = get_function_source(func)
        metrics = metrics_calculator.compute_metrics(code, eta2_star)

        print(f"η₂* (смысловые параметры): {metrics['eta2_star']}")
        print(f"η₁ (уникальные операторы): {metrics['eta1']}")
        print(f"η₂ (уникальные операнды): {metrics['eta2']}")
        print(f"η (словарь): {metrics['eta']}")
        print(f"N₁ (вхождения операторов): {metrics['N1']}")
        print(f"N₂ (вхождения операндов): {metrics['N2']}")
        print(f"N (длина реализации): {metrics['N']}")
        print(f"Ń (предсказанная длина): {metrics['N_hat']:.2f}")
        print(f"V* (потенциальный объём): {metrics['V_star']:.2f}")
        print(f"V (объём реализации): {metrics['V']:.2f}")
        print(f"L (уровень через V*): {metrics['L']:.4f}")
        print(f"L̂ (уровень по реализации): {metrics['L_hat']:.4f}")
        print(f"I (интеллектуальное содержание): {metrics['I']:.2f}")
        print(f"T̂₁ (время по V*): {metrics['T1']:.2f} сек")
        print(f"T̂₂ (время по Ń): {metrics['T2']:.2f} сек")
        print(f"T̂₃ (время по реализации): {metrics['T3']:.2f} сек")

        all_L_hat.append(metrics["L_hat"])
        all_V.append(metrics["V"])

    n = len(all_L_hat)
    if n > 0:
        lambda1 = sum(l_hat * v for l_hat, v in zip(all_L_hat, all_V)) / (2 * n)
        lambda2 = sum(v * v for v in all_V) / (2 * n)

        print("\n" + "=" * 100)
        print("СРЕДНИЕ ЗНАЧЕНИЯ УРОВНЕЙ ЯЗЫКА ПРОГРАММИРОВАНИЯ:")
        print(f"λ₁ = (Σ L̂ᵢ·Vᵢ) / (2·n) = {lambda1:.2f}")
        print(f"λ₂ = (Σ Vᵢ²) / (2·n) = {lambda2:.2f}")
        print("=" * 100)


if __name__ == "__main__":
    main()
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>

#define S 18.0

typedef struct { int val, idx; } Min1D;
typedef struct { int val, row, col; } Min2D;

typedef struct {
    int eta1, eta2, eta, N1, N2, N;
    double N_hat, V_star, V, L, L_hat, I, T1, T2, T3;
} Metrics;

Min1D task1_find_min(int* arr, int n) {
    Min1D res = {arr[0], 0};
    int i;
    for (i = 1; i < n; i++) {
        if (arr[i] < res.val) {
            res.val = arr[i];
            res.idx = i;
        }
    }
    return res;
}

void task2_bubble_sort(int* arr, int n) {
    int i, j, tmp;
    for (i = 0; i < n; i++) {
        for (j = 0; j < n - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {
                tmp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = tmp;
            }
        }
    }
}

int task3_binary_search(int* arr, int n, int target) {
    int left = 0, right = n - 1;
    while (left <= right) {
        int mid = left + (right - left) / 2;
        if (arr[mid] == target) return mid;
        if (arr[mid] < target) left = mid + 1;
        else right = mid - 1;
    }
    return -1;
}

Min2D task4_find_min_2d(int** matrix, int rows, int* cols) {
```

```
    Min2D res = {matrix[0][0], 0, 0};
    int i, j;
    for (i = 0; i < rows; i++) {
        for (j = 0; j < cols[i]; j++) {
            if (matrix[i][j] < res.val) {
                res.val = matrix[i][j];
                res.row = i;
                res.col = j;
            }
        }
    }
    return res;
}

void task5_reverse_array(int* arr, int n) {
    int i, tmp;
    for (i = 0; i < n / 2; i++) {
        tmp = arr[i];
        arr[i] = arr[n - 1 - i];
        arr[n - 1 - i] = tmp;
    }
}

void task6_cyclic_shift_left(int* arr, int n, int k) {
    k %= n;
    int* tmp = malloc(n * sizeof(int));
    int i;
    for (i = 0; i < n; i++) tmp[i] = arr[(i + k) % n];
    for (i = 0; i < n; i++) arr[i] = tmp[i];
    free(tmp);
}

void task7_replace_value(int* arr, int n, int old_val, int new_val) {
    int i;
    for (i = 0; i < n; i++) {
        if (arr[i] == old_val) arr[i] = new_val;
    }
}

Metrics compute_metrics(int eta1, int eta2, int N1, int N2, int eta2_star) {
    Metrics m;
    int eta = eta1 + eta2;
    int N = N1 + N2;

    m.eta1 = eta1;
    m.eta2 = eta2;
    m.eta = eta;
    m.N1 = N1;
    m.N2 = N2;
    m.N = N;

    m.N_hat = (eta1 > 0 ? eta1 * log2(eta1) : 0) + (eta2 > 0 ? eta2 * log2(eta2) : 0);
```

```
    m.V_star = (2 + eta2_star) * log2(2 + eta2_star);
    m.V = (eta > 0 && N > 0) ? N * log2(eta) : 0;
    m.L = (m.V > 0) ? m.V_star / m.V : 0;
    m.L_hat = (eta1 > 0 && N2 > 0) ? (2.0 / eta1) * (eta2 / (double)N2) : 0;
    m.I = (eta1 > 0 && N1 > 0 && eta > 0) ? (2.0 / eta1) * (N2 / (double)N1) * eta *
log2(eta) : 0;
    m.T1 = m.V_star / S;
    m.T2 = (m.N_hat * (eta1 * log2(eta2) + eta2 * log2(eta1))) / (2 * S);
    m.T3 = (N1 * N2 * log2(eta)) / (2 * S);

    return m;
}

const char* task_names[] = {
    "1. Минимальный элемент одномерного массива и его индекс",
    "2. Сортировка пузырьком",
    "3. Бинарный поиск",
    "4. Минимальный элемент двумерного массива",
    "5. Перестановка в обратном порядке",
    "6. Циклический сдвиг влево на k позиций",
    "7. Замена всех вхождений значения"
};

int eta2_stars[] = {3, 3, 3, 4, 2, 3, 4};

int main() {

printf("================================================================================
====================\n");
    printf("ЛАБОРАТОРНАЯ РАБОТА №12: МЕТРИКИ ХОЛСТЕДА (C)\n");

printf("================================================================================
====================\n");

    int task_params[7][3] = {
        {7, 6, 13}, {16, 32, 24}, {14, 24, 20}, {17, 36, 28},
        {11, 12, 10}, {17, 28, 22}, {12, 14, 12}
    };

    double sum_l_hat_v = 0, sum_v_sq = 0;

    for (int i = 0; i < 7; i++) {
        Metrics m = compute_metrics(task_params[i][0], task_params[i][1],
                                    task_params[i][0], task_params[i][1],
eta2_stars[i]);

        printf("\n--- ЗАДАЧА %d: %s ---\n", i+1, task_names[i]);
        printf("η₂* (смысловые параметры): %d\n", eta2_stars[i]);
        printf("η₁ (уникальные операторы): %d\n", m.eta1);
        printf("η₂ (уникальные операнды): %d\n", m.eta2);
        printf("η (словарь): %d\n", m.eta);
        printf("N₁ (вхождения операторов): %d\n", m.N1);
```

```
        printf("N₂ (вхождения операндов): %d\n", m.N2);
        printf("N (длина реализации): %d\n", m.N);
        printf("Ń (предсказанная длина): %.2f\n", m.N_hat);
        printf("V* (потенциальный объём): %.2f\n", m.V_star);
        printf("V (объём реализации): %.2f\n", m.V);
        printf("L (уровень через V*): %.4f\n", m.L);
        printf("L̂ (уровень по реализации): %.4f\n", m.L_hat);
        printf("I (интеллектуальное содержание): %.2f\n", m.I);
        printf("T̂₁ (время по V*): %.2f сек\n", m.T1);
        printf("T̂₂ (время по Ń): %.2f сек\n", m.T2);
        printf("T̂₃ (время по реализации): %.2f сек\n", m.T3);

        sum_l_hat_v += m.L_hat * m.V;
        sum_v_sq += m.V * m.V;
    }

    double lambda1 = sum_l_hat_v / 14.0;
    double lambda2 = sum_v_sq / 14.0;


printf("\n=========================================================================
======================\n");
    printf("СРЕДНИЕ ЗНАЧЕНИЯ УРОВНЕЙ ЯЗЫКА ПРОГРАММИРОВАНИЯ:\n");
    printf("λ₁ = (Σ L̂ᵢ·Vᵢ) / (2·n) = %.2f (Python: 5.52)\n", lambda1);
    printf("λ₂ = (Σ Vᵢ²) / (2·n) = %.2f (Python: 29906.42)\n", lambda2);

printf("=========================================================================
====================\n");

    return 0;
}
```

Результат

η_2^*	η_1	η_2	η	N_1	N_2	N	$\hat{N}$	V^*	V	L	$\hat{L}$	I	$\hat{T}_1$	$\hat{T}_2$	$\hat{T}_3$
3	14	10	24	29	23	52	86.52	11.61	238.42	0.0487	0.0621	12.47	0.64	203.28	61.55
2	14	8	22	42	30	72	77.30	8.00	321.08	0.0249	0.0381	10.01	0.44	155.59	105.00
3	18	9	27	33	26	59	103.59	11.61	280.54	0.0414	0.0385	11.24	0.64	272.17	75.55
4	15	11	26	51	36	87	96.66	15.51	408.94	0.0379	0.0407	11.50	0.86	254.71	176.43
2	5	2	7	6	2	8	13.61	8.00	22.46	0.3562	0.4000	2.62	0.44	3.65	0.33
3	11	3	14	16	10	26	42.81	11.61	98.99	0.1173	0.0545	6.06	0.64	33.07	7.04
4	8	4	12	8	6	14	32.00	15.51	50.19	0.3090	0.1667	8.07	0.86	24.89	2.67

Сводные λ (Python)
$\lambda_1 = 5.52$.
$\lambda_2 = 29906.42$

Анализ полученных результатов (Python):
Сложность реализованных задач различается: наибольшие объёмы V и длины N получились у поиска минимума в матрице и пузырька из-за вложенных циклов и большего числа операторов/операндов, что приводит к низкому уровню L≈0.038–0.042 при высоком V, указывая на высокую алгоритмическую трудоёмкость реализации в выбранной нотации.

Простые операции, такие как разворот массива и замена значений по условию, показали минимальные N и высокие уровни L≈0.1667–0.4000 при малых объёмах, что согласуется с их линейной структурой и малым словарём η.

Прогнозируемая длина $\hat{N}$ систематически выше фактической для более сложных функций, что обычно связано с усреднённой природой оценки Холстеда и особенностями токенизации Python, где часть семантики упакована в синтаксис с минимальным количеством токенов.

Итоговые показатели языка λ_1 и λ_2 отражают агрегированную «уровневость» и суммарные объёмы по набору задач: умеренное λ_1 при очень большом λ_2 указывает, что совокупная трудоёмкость доминирует за счёт задач с большими V, хотя относительный уровень реализации не экстремально низкий в среднем.

Сводные λ (C)

$\lambda_1 = 12.84$
$\lambda_2 = 20924.63$

Анализ полученных результатов (C):

Сложность реализованных задач различается: наибольшие объёмы V и длины N получились у поиска минимума в матрице и пузырька из-за вложенных циклов и большего числа операторов/операндов, что приводит к низкому уровню L≈0.043–0.058 при высоком V, указывая на высокую алгоритмическую трудоёмкость реализации в C-нотации.

Простые операции, такие как разворот массива и замена значений по условию, показали минимальные N и высокие уровни L≈0.127–0.182 при малых объёмах, что согласуется с их линейной структурой и малым словарем η.

Прогнозируемая длина $\acute{N}$ систематически выше фактической для более сложных функций, что обычно связано с усреднённой природой оценки Холстеда и особенностями токенизации C, где явные операторы и переменные увеличивают словарь η по сравнению с Python.

Итоговые показатели языка λ_1 и λ_2 отражают агрегированную «уровневость» и суммарные объёмы по набору задач: более высокое λ_1 при меньшем λ_2 указывает, что C требует большего словаря η и явных конструкций, но даёт меньшую суммарную трудоёмкость за счёт компактных токенов по сравнению с Python.