Федеральное государственное бюджетное образовательное учреждение высшего образования
«Сибирский государственный университет телекоммуникаций и информатики»
(СибГУТИ)

Институт информатики и вычислительной техники

09.03.01 "Информатика и вычислительная техника"
профиль "Программное обеспечение средств
вычислительной техники и автоматизированных систем"

Кафедра прикладной математики и кибернетики

Современные технологии программирования

Лабораторная работа №12

Вычисление метрических характеристик реализаций алгоритмов

Выполнил:

студент гр.ИП-213 Дмитриев Антон Александрович
ФИО студента

«__» _____ 2025 г.

Проверил:

Преподаватель

ФИО преподавателя

«__» _____ 2025 г. Оценка_____

Новосибирск 2025 г.

1. **Текст программы**

C++:

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

// 1. Поиск минимального элемента в одномерном массиве
void findMin1D(int arr[], int size, int& minValue, int& minIndex) {
    if (size == 0) {
        minValue = -1;
        minIndex = -1;
        return;
    }

    minValue = arr[0];
    minIndex = 0;

    for (int i = 1; i < size; i++) {
        if (arr[i] < minValue) {
            minValue = arr[i];
            minIndex = i;
        }
    }
}

// 2. Сортировка пузырьком
void bubbleSort(int arr[], int size) {
    for (int i = 0; i < size - 1; i++) {
        for (int j = 0; j < size - i - 1; j++) {
```

```
            if (arr[j] > arr[j + 1]) {
                swap(arr[j], arr[j + 1]);
            }
        }
    }
}

// 3. Бинарный поиск
int binarySearch(int arr[], int size, int target) {
    int left = 0, right = size - 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; // элемент не найден
}

// 4. Поиск минимального элемента в двумерном массиве
void findMin2D(vector<vector<int>>& matrix, int& minValue, int& minRow, int& minCol) {
    if (matrix.empty() || matrix[0].empty()) {
```

```
        minValue = -1;
        minRow = -1;
        minCol = -1;
        return;
    }

    minValue = matrix[0][0];
    minRow = 0;
    minCol = 0;

    for (int i = 0; i < matrix.size(); i++) {
        for (int j = 0; j < matrix[i].size(); j++) {
            if (matrix[i][j] < minValue) {
                minValue = matrix[i][j];
                minRow = i;
                minCol = j;
            }
        }
    }
}

// 5. Перестановка элементов в обратном порядке
void reverseArray(int arr[], int size) {
    for (int i = 0; i < size / 2; i++) {
        swap(arr[i], arr[size - i - 1]);
    }
}

// 6. Циклический сдвиг влево
void leftRotate(int arr[], int size, int positions) {
```

```
    if (size == 0) return;

    positions = positions % size; // обработка случая, когда positions > size
    if (positions == 0) return;

    // Временный массив для хранения первых positions элементов
    int* temp = new int[positions];

    // Сохраняем первые positions элементов
    for (int i = 0; i < positions; i++) {
        temp[i] = arr[i];
    }

    // Сдвигаем остальные элементы влево
    for (int i = positions; i < size; i++) {
        arr[i - positions] = arr[i];
    }

    // Восстанавливаем сохраненные элементы в конец
    for (int i = 0; i < positions; i++) {
        arr[size - positions + i] = temp[i];
    }

    delete[] temp;
}

// 7. Замена всех вхождений значения
void replaceAll(int arr[], int size, int oldValue, int newValue) {
    for (int i = 0; i < size; i++) {
        if (arr[i] == oldValue) {
```

```
            arr[i] = newValue;
        }
    }
}

// Вспомогательная функция для печати массива
void printArray(int arr[], int size) {
    for (int i = 0; i < size; i++) {
        cout << arr[i] << " ";
    }
    cout << endl;
}
```

Python:

```
from typing import List, Tuple, Optional

# 1. Поиск минимального элемента в одномерном массиве
def find_min_1d(arr: List[int]) -> Tuple[Optional[int], Optional[int]]:
    """Возвращает (минимальное значение, индекс) или (None, None) если массив пуст"""
    if not arr:
        return None, None

    min_value = arr[0]
    min_index = 0

    for i in range(1, len(arr)):
        if arr[i] < min_value:
            min_value = arr[i]
            min_index = i
```

```
    return min_value, min_index


# 2. Сортировка пузырьком
def bubble_sort(arr: List[int]) -> None:
    n = len(arr)
    for i in range(n - 1):
        swapped = False
        for j in range(n - i - 1):
            if arr[j] > arr[j + 1]:
                arr[j], arr[j + 1] = arr[j + 1], arr[j]
                swapped = True
        if not swapped:
            break


# 3. Бинарный поиск
def binary_search(arr: List[int], target: int) -> int:
    left, right = 0, len(arr) - 1

    while left <= right:
        mid = left + (right - left) // 2

        if arr[mid] == target:
            return mid
        elif arr[mid] < target:
            left = mid + 1
        else:
            right = mid - 1

    return -1  # элемент не найден
```

```
# 4. Поиск минимального элемента в двумерном массиве
def find_min_2d(matrix: List[List[int]]) -> Tuple[Optional[int], Optional[int], Optional[int]]:
    """Возвращает (минимальное значение, строка, столбец)"""
    if not matrix or not matrix[0]:
        return None, None, None

    min_value = matrix[0][0]
    min_row, min_col = 0, 0

    for i in range(len(matrix)):
        for j in range(len(matrix[i])):
            if matrix[i][j] < min_value:
                min_value = matrix[i][j]
                min_row, min_col = i, j

    return min_value, min_row, min_col

# 5. Перестановка элементов в обратном порядке
def reverse_array(arr: List[int]) -> None:
    left, right = 0, len(arr) - 1
    while left < right:
        arr[left], arr[right] = arr[right], arr[left]
        left += 1
        right -= 1

# 6. Циклический сдвиг влево
def left_rotate(arr: List[int], positions: int) -> None:
    if not arr:
        return
```

```
    n = len(arr)
    positions = positions % n  # обработка случая, когда positions > n

    if positions == 0:
        return

def replace_all(arr: List[int], old_value: int, new_value: int) -> None:
    for i in range(len(arr)):
        if arr[i] == old_value:
            arr[i] = new_value
```

2. Метрические характеристики реализации и алгоритма C++:

```
void findMin1D(int arr[], int size, int& minValue, int& minIndex) {
   if (size == 0) {
      minValue = -1;
      minIndex = -1;
      return;
   }

   minValue = arr[0];
   minIndex = 0;

   for (int i = 1; i < size; i++) {
      if (arr[i] < minValue) {
         minValue = arr[i];
         minIndex = i;
      }
   }
}
```

Анализ для C++ версии:

Операторы (η_1):

- void (тип возвращаемого значения)
- findMin1D (имя функции)
- int (тип параметра)
- [] (оператор массива)
- & (ссылка)
- if
- ==
- =
- -
- return
- for
- ; (разделитель)

- <
- ++ (инкремент)
- {} (блоки)

Операнды (η_2):

- arr
- size
- minValue
- minIndex
- 0
- 1
- i

Расчет метрик для C++ findMin1D:

$\eta^*_2 = 4$ (входные: arr, size; выходные: minValue, minIndex)

$\eta_1 = 15$

$\eta_2 = 7$

$\eta = \eta_1 + \eta_2 = 22$

$N_1 = 38$ (общее число операторов)

$N_2 = 27$ (общее число операндов)

$N = N_1 + N_2 = 65$

$\hat{N} = \eta_1 \cdot \log_2\eta_1 + \eta_2 \cdot \log_2\eta_2 = 15 \cdot \log_2 15 + 7 \cdot \log_2 7 = 15 \cdot 3.9069 + 7 \cdot 2.8074 = 58.6035 + 19.6518 = 78.2553$

$V^* = (2 + \eta^*_2) \cdot \log_2(2 + \eta^*_2) = (2 + 4) \cdot \log_2 6 = 6 \cdot 2.5850 = 15.51$

$V = N \cdot \log_2\eta = 65 \cdot \log_2 22 = 65 \cdot 4.4594 = 289.861$

$L = V^*/V = 15.51/289.861 = 0.0535$

$\hat{L} = (2 \cdot \eta_2)/(\eta_1 \cdot N_2) = (2 \cdot 7)/(15 \cdot 27) = 14/405 = 0.0346$

$I = \hat{L} \cdot V = 0.0346 \cdot 289.861 = 10.031$

$S = 18$ (стандартная константа Страуда)

$T_1 = V^{*2}/(2 \cdot S \cdot V) = 15.51^2/(2 \cdot 18 \cdot 289.861) = 240.5601/(10435.0) = 0.0231$ секунд

$\hat{T}_2 = (\eta_1 \cdot N \cdot (\eta_1 \cdot \log_2\eta_1 + \eta_2 \cdot \log_2\eta_2) \cdot \log_2\eta)/(2 \cdot S \cdot \eta_2 \cdot N) = (15 \cdot 65 \cdot 78.2553 \cdot 4.4594)/(2 \cdot 18 \cdot 7 \cdot 65) = (15 \cdot 78.2553 \cdot 4.4594)/(2 \cdot 18 \cdot 7) = 5236.8/252 = 20.78$ секунд

$\hat{T}_3 = (\eta_1 \cdot N \cdot N \cdot \log_2\eta)/(2 \cdot S \cdot \eta_2 \cdot N) = (15 \cdot 65 \cdot 65 \cdot 4.4594)/(2 \cdot 18 \cdot 7 \cdot 65) = (15 \cdot 65 \cdot 4.4594)/(2 \cdot 18 \cdot 7) = 4347.9/252 = 17.26$ секунд

	η^*_2	η_1	η_2	η	N_1	N_2	N	$\hat{N}$	V^*	V	L	$\hat{L}$	I	$\hat{T}_1$	$\hat{T}_2$	$\hat{T}_3$	λ_1	λ_2
findMin1D	**4**	**15**	**7**	**22**	**38**	**27**	**65**	**78.3**	**15.5**	**290**	**0.054**	**0.035**	**10.0**	**0.023**	**20.8**	**17.3**	**0.011**	**0.054**
bubbleSort	**2**	**18**	**8**	**26**	**58**	**42**	**100**	**99.8**	**8.0**	**458**	**0.017**	**0.021**	**9.6**	**0.007**	**39.2**	**39.2**	**0.008**	**0.017**
binarySearch	**3**	**17**	**9**	**26**	**42**	**31**	**73**	**92.4**	**10.8**	**334**	**0.032**	**0.040**	**13.4**	**0.017**	**24.0**	**23.1**	**0.015**	**0.032**
findMin2D	**4**	**18**	**10**	**28**	**49**	**40**	**89**	**110.9**	**15.5**	**408**	**0.038**	**0.028**	**11.4**	**0.015**	**30.8**	**29.3**	**0.017**	**0.038**
reverseArray	**2**	**11**	**6**	**17**	**27**	**17**	**44**	**46.5**	**8.0**	**188**	**0.043**	**0.065**	**12.2**	**0.017**	**9.4**	**8.9**	**0.021**	**0.043**
leftRotate	**3**	**25**	**11**	**36**	**78**	**56**	**134**	**143.8**	**10.8**	**614**	**0.018**	**0.016**	**9.8**	**0.010**	**52.8**	**52.8**	**0.009**	**0.018**
replaceAll	**3**	**11**	**7**	**18**	**24**	**18**	**42**	**47.0**	**10.8**	**179**	**0.060**	**0.071**	**12.7**	**0.033**	**7.2**	**6.8**	**0.030**	**0.060**
Среднее	**3.0**	**16.4**	**8.3**	**24.7**	**45.1**	**33.0**	**78.1**	**88.1**	**11.4**	**353**	**0.038**	**0.039**	**11.3**	**0.018**	**26.3**	**25.4**	**0.016**	**0.038**

Python:

```
def find_min_1d(arr: List[int]) -> Tuple[Optional[int], Optional[int]]:
    if not arr:
        return None, None

    min_value = arr[0]
    min_index = 0

    for i in range(1, len(arr)):
        if arr[i] < min_value:
            min_value = arr[i]
            min_index = i

    return min_value, min_index
```

Анализ для Python версии:

Операторы (η_1):

- def
- :
- ->(аннотация возвращаемого значения)
- if
- not
- return
- =
- for
- in
- range
- len
- <
- [] (индексация)

Операнды (η_2):

- find_min_1d
- arr
- List
- int
- Tuple
- Optional
- None
- min_value
- min_index
- 0
- 1
- i

Расчет метрик для Python find_min_1d:

$\eta^*_2 = 1$ (входные: arr; выходные: кортеж через return)

$\eta_1 = 13$

$\eta_2 = 12$

$\eta = \eta_1 + \eta_2 = 25$

$N_1 = 28$

$N_2 = 26$

$N = 54$

$\hat{N} = 13 \cdot \log_2 13 + 12 \cdot \log_2 12 = 13 \cdot 3.7004 + 12 \cdot 3.5850 = 48.1052 + 43.0200 = 91.1252$

$V^* = (2 + 1) \cdot \log_2(3) = 3 \cdot 1.5850 = 4.755$

$V = 54 \cdot \log_2 25 = 54 \cdot 4.6439 = 250.771$

$L = 4.755/250.771 = 0.0190$

$\hat{L} = (2 \cdot 12)/(13 \cdot 26) = 24/338 = 0.0710$

$I = 0.0710 \cdot 250.771 = 17.805$

$\hat{T}_1 = 4.755^2/(2 \cdot 18 \cdot 250.771) = 22.61/(9027.8) = 0.0025$ секунд

$\hat{T}_2 = (13 \cdot 54 \cdot 91.1252 \cdot 4.6439)/(2 \cdot 18 \cdot 12 \cdot 54) = (13 \cdot 91.1252 \cdot 4.6439)/(2 \cdot 18 \cdot 12) = 5500.8/432 = 12.73$ секунд

$\hat{T}_3 = (13 \cdot 54 \cdot 54 \cdot 4.6439)/(2 \cdot 18 \cdot 12 \cdot 54) = (13 \cdot 54 \cdot 4.6439)/(2 \cdot 18 \cdot 12) = 3261.6/432 = 7.55$ секунд

	η^*_2	η_1	η_2	η	N_1	N_2	N	$\hat{N}$	V^*	V	L	$\hat{L}$	I	$\hat{T}_1$	$\hat{T}_2$	$\hat{T}_3$	λ_1	λ_2
find_min_1d	**1**	**13**	**12**	**25**	**28**	**26**	**54**	**91.1**	**4.76**	**251**	**0.019**	**0.071**	**17.8**	**0.0025**	**12.7**	**7.55**	**0.008**	**0.019**
bubble_sort	**1**	**14**	**11**	**25**	**35**	**29**	**64**	**85.4**	**4.76**	**296**	**0.016**	**0.054**	**16.0**	**0.0019**	**13.4**	**10.15**	**0.007**	**0.016**
binary_search	**2**	**14**	**13**	**27**	**31**	**27**	**58**	**93.8**	**8.00**	**272**	**0.029**	**0.069**	**18.8**	**0.0118**	**12.0**	**9.36**	**0.014**	**0.029**
find_min_2d	**1**	**15**	**14**	**29**	**36**	**31**	**67**	**108.7**	**4.76**	**322**	**0.015**	**0.067**	**21.6**	**0.0018**	**16.0**	**13.44**	**0.007**	**0.015**
reverse_array	**1**	**9**	**9**	**18**	**17**	**15**	**32**	**52.2**	**4.76**	**137**	**0.035**	**0.133**	**18.2**	**0.0083**	**4.8**	**4.27**	**0.018**	**0.035**
left_rotate	**2**	**8**	**8**	**16**	**14**	**13**	**27**	**42.1**	**8.00**	**108**	**0.074**	**0.154**	**16.6**	**0.0296**	**3.8**	**3.78**	**0.037**	**0.074**
replace_all	**2**	**7**	**8**	**15**	**12**	**11**	**23**	**37.8**	**8.00**	**89**	**0.090**	**0.182**	**16.2**	**0.0360**	**2.7**	**2.67**	**0.045**	**0.090**
Среднее	**1.4**	**11.4**	**10.7**	**22.1**	**24.7**	**21.7**	**46.4**	**73.0**	**6.18**	**211**	**0.040**	**0.104**	**17.9**	**0.013**	**9.3**	**7.3**	**0.019**	**0.04**

3. Анализ полученных результатов

1. **Сложность программ**:
   - C++ программы в среднем длиннее (N=78.1 против 46.4)
   - Python имеет более высокий уровень программы $\hat{L}$ (0.104 против 0.039), что указывает на более высокий уровень абстракции
2. **Эффективность языков**:
   - Python имеет меньший объем реализации V (211 против 353)
   - Python имеет большее интеллектуальное содержание I (17.9 против 11.3)
3. **Время разработки**:
   - Прогнозируемое время разработки $\hat{T}_2$ и $\hat{T}_3$ меньше для Python (9.3 и 7.3 секунд против 26.3 и 25.4 секунд для C++)
4. **Уровни языков**:
   - λ_1 (средний уровень через $\hat{L} \cdot V$): Python 0.019, C++ 0.016
   - λ_2 (средний уровень через L): Python 0.040, C++ 0.038
   - Python показывает немного более высокие значения, что соответствует его более высокоуровневой природе
5. **Параметры ввода-вывода**:
   - Python имеет меньшее η^*_2 (1.4 против 3.0), что связано с использованием возвращаемых значений вместо выходных параметров