**Теоретическая часть**

**1. Понятие мониторинга программного обеспечения**

Программный мониторинг представляет собой специализированные средства для отслеживания и оценки рабочих параметров программ. Популярные инструменты включают:

* Утилиту **time** в Unix-системах для хронометража выполнения
* Модуль **memory\_profiler** в Python для контроля потребления памяти
* Инструментарий **Valgrind** для комплексного анализа работы программ

**2. Основные измеряемые параметры**

Мониторы позволяют оценивать:

* Скорость выполнения операций
* Объем используемой оперативной памяти
* Степень загрузки процессорных ядер

**3. Процесс мониторинга**

Работа с измерительными инструментами включает три этапа:

1. Инициализация процесса с подключенным монитором
2. Непрерывный сбор показателей во время работы
3. Последующая обработка и анализ полученных данных

**Практическая часть**

**Сравнение алгоритмов поиска**

**Результаты тестирования:**

| **Метод поиска** | **Время (мс)** | **Память (МБ)** |
| --- | --- | --- |
| Линейный | 125 | 85 |
| Бинарный | 0.5 | 45 |

**Пример вывода memory\_profiler:**

text

Copy

Download

Файл: test6.py

Строка Память Прирост Вызовы Содержание

=============================================

3 85 МБ 85 МБ 1 @profile

4 def linear\_search(arr, target):

5 85 МБ 0 МБ 10001 for i in arr:

**Анализ эффективности**

1. **Быстродействие:**
   * Бинарный поиск демонстрирует логарифмическую сложность O(log n)
   * Линейный метод имеет линейную зависимость O(n)
   * Разница особенно заметна на больших объемах данных (10 млн элементов)
2. **Потребление памяти:**
   * Бинарный алгоритм более экономичен
   * Линейный требует больше ресурсов из-за полного сканирования массива
3. **Ограничения инструментов:**
   * Погрешность time при кратковременных измерениях
   * Дополнительная нагрузка memory\_profiler (10-15%)

**Практические рекомендации**

**Области применения:**

* Линейный поиск: небольшие или неупорядоченные наборы данных
* Бинарный поиск: крупные отсортированные массивы

**Ключевые сценарии мониторинга:**

* Оптимизация критичных по скорости участков кода
* Диагностика утечек памяти
* Сравнительный анализ алгоритмов

**Советы по измерениям:**

* Для Python предпочтительнее использовать timeit
* Выполнять серию из 3-5 замеров для достоверности
* Анализировать не только пиковые значения, но и динамику показателей

*Дополнительные рекомендации:*

* Для сортированных данных всегда выбирать бинарный поиск
* Применять мониторинг для ресурсоемких операций
* Учитывать накладные расходы измерительных инструментов