# Повторение

1. **Алгоритмы**

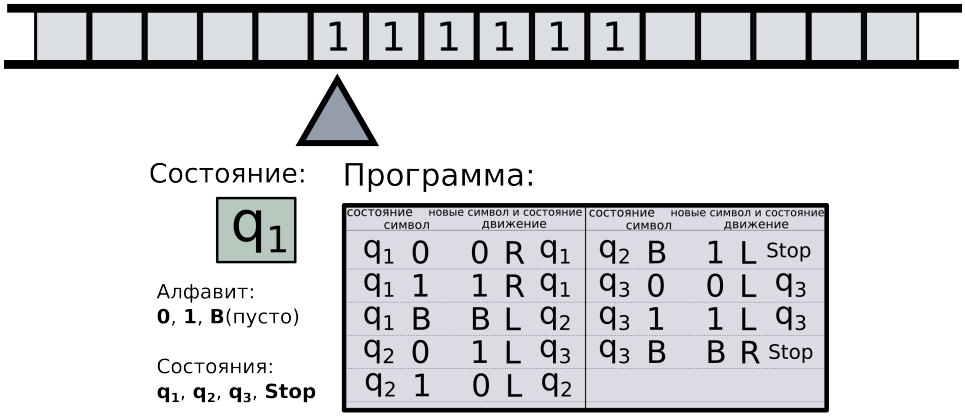
   (a) Заполните таблицу:

| Алгоритм | Сложность(в среднем) |
|---|---|
| Сортировка пузырьком | $O(n^2)$ |
| Сортировка вставками | |
| Быстрая сортировка | |
| Цифровая сортировка | |
| Простейший алгоритм перемножения матриц nxn | |
| Добавление элемента в начало массива | |
| Добавление элемента в начало связного списка | |

   (b) Предположим, что лучший алгоритм для задачи имеет сложность $O(N^2)$. К какому классу принадлежит данная задача (P или NP). То же самое для сложностей:

   - $O(N^{10} + 5N^5)$.
   - $O(log(N))$.
   - $O(2^N)$

   (c) Машина Тьюринга: проведите вычисления для следующего примера:




   (d) Напишите программу для машины Тьюринга, которая умножает двоичное число на 2.

   (e) Класс NP это:

   - Класс задач, которые нельзя решить за полиномиальное время на детерминированной машине Тьюринга.
   - Класс задач, которые можно решить за полиномиальное время на недетерминированной машине Тьюринга.
   - Класс алгоритмов, которые нельзя выполнить за полиномиальное время на детерминированной машине Тьюринга.
   - Класс алгоритмов, которые можно выполнить за полиномиальное время на недетерминированной машине Тьюринга.

2. **Основы командной строки**

- `pwd` – печать текущей директории
- `ls <имя директории>` – просмотр директории
- `cd <имя директории>` – переход в другую директорию
- `mkdir <имя директории>` – создать директорию
- `subl <имя файла>` или `nano <имя файла>` – открыть файл в соответствующем текстовом редакторе.
- `.` – текущая папка `..` – родительская папка `/` – корневая папка (аналог C: в windows) тильда – домашняя папка.
- `top` – диспетчер задач (выйти Ctrl-C).
- `gcc <code.c>` – компиляция кода из файла `code.c`
- `./<имя программы>` – запуск программы

(a) Откройте терминал и перейдите в домашнюю папку (`/home-local/student`).

(b) Создайте свою папку в этой директории и перейдите в неё.

(c) Создайте простейшую программу и скомпилируйте её, используя gcc.

3. **Основы C: переменные, массивы, функции, структуры, указатели**

(a) Объявите переменную типа double и присвойте ей значение 2.718281828.

(b) Объявите статический массив переменных типа int размера 6 элементов и инициализируйте его следующими значениями: 4, 8, 15, 16, 23, 42. (решение – 1 строка)

(c) Создайте динамический массив переменных типа int размера 6 элементов и инициализируйте его теми же значениями. (используйте malloc() и free() из `stdlib.h`).

(d) Создайте функцию `float geometric_mean(float a, float b)`, которая вычисляет геометрическое среднее. Не забудьте подключить математическую библиотеку: `math.h`, а также добавить опцию `-lm` при компиляции.

(e) Опишите структуру `Complex`, описывающую комплексное число.

(f) Напишите функцию `Complex complex_add(Complex a, Complex b)`, которая складывает комплексные числа. Проверьте работу функции.

(g) Напишите функцию `void complex_sqr(Complex* pc)`, которая возводит комплексное число в квадрат. Проверьте работу функции на таком тесте: $(3 + 2i)^2 = 5 + 12i$.

(h) Создайте новый файл `complex.h` и перенесите всё описание `Complex` в этот файл. Подключите этот файл к основному с помощью `#include "complex.h"`.

(i) Добавьте новые функции для работы со структурой `Complex` в файл `complex.h`.