Министерство образования и науки Российской Федерации федеральное государственное
автономное образовательное учреждение высшего образования
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Факультет «Программной инженерии и компьютерной техники.»

# Алгоритмы и структуры данных

## Лабораторная работа №1
## Базовые задачи

**Выполнил**
Григорьев Давид Владимирович
Группа: P3215
**Преподаватели**
Косяков Михаил Сергеевич
Тараканов Денис Сергеевич

Санкт-Петербург 2025

# Содержание

# 1 Задача A. Агроном-любитель

## 1.1 Пояснение к примененному алгоритму

Time complexity: O(n) Space complexity: O(1)

## 1.2 Код алгоритма

```cpp
#include <iostream>

int main() {
  size_t input_size = 0;
  std::cin >> input_size;

  size_t max_count = 0;
  size_t max_staring_index = 1;
  size_t max_ending_index = 0;

  size_t current_repeating_count = 1;
  size_t current_max_count = 0;
  size_t last_begining_index = 1;

  int last_num = 0;

  for (size_t i = 1; i < input_size + 1; i++) {
    int new_num = 0;
    std::cin >> new_num;

    if (new_num == last_num) {
      current_repeating_count++;
    } else {
      current_repeating_count = 1;
    }

    last_num = new_num;
    if (current_repeating_count < 3) {
      current_max_count++;

      if (current_max_count > max_count) {
        max_count = current_max_count;
        max_staring_index = last_begining_index;
        max_ending_index = i;
      }
    } else {
      last_begining_index = i - 1;
      current_max_count = 2;
      current_repeating_count = 2;
    }
  }

  std::cout << max_staring_index << " " << max_ending_index << '\n';

  return 0;
}
```

# 2 Задача B. Зоопарк Глеба

## 2.1 Пояснение к примененному алгоритму

Time complexity: O(n) Space complexity: O(n)

## 2.2 Код алгоритма

```cpp
#include <cctype>
#include <iostream>
#include <map>
#include <stack>
#include <string>

int main() {
  std::string input_string;
  std::cin >> input_string;

  std::stack<char> brackets;
  std::map<int, int> trap_index_to_animal_index;

  std::stack<int> animal_index_stack;
  std::stack<int> trap_index_stack;

  int last_animal_index = 1;
  int last_trap_index = 1;

  for (char character : input_string) {
    if (std::islower(character) != 0) {
      animal_index_stack.push(last_animal_index++);
    } else {
      trap_index_stack.push(last_trap_index++);
    }

    if (brackets.empty()) {
      brackets.push(character);
      continue;
    }

    const char last_char = brackets.top();
    if (std::tolower(character) == std::tolower(last_char) && (character != last_char))
      trap_index_to_animal_index[trap_index_stack.top()] = animal_index_stack.top();
      animal_index_stack.pop();
      trap_index_stack.pop();

      brackets.pop();
    } else {
      brackets.push(character);
    }
  }

  if (!brackets.empty()) {
    std::cout << "Impossible" << '\n';
    return 0;
  }
```

```
  std::cout << "Possible" << '\n';
  for (auto& pair : trap_index_to_animal_index) {
    std::cout << pair.second << '\n';
  }

  return 0;
}
```

# 3 Задача C. Конфигурационный файл

## 3.1 Пояснение к примененному алгоритму

Time complexity: O(n) Space complexity: O(n)
Хотя насчет времени я не уверен

## 3.2 Код алгоритма

```cpp
#include <algorithm>
#include <cctype>
#include <cstdlib>
#include <iostream>
#include <stack>
#include <string>
#include <unordered_map>
#include <unordered_set>

int main() {
  std::string last_string;

  std::unordered_map<std::string, std::stack<int>> var_name_to_value_history;

  std::stack<std::unordered_set<std::string>> used_vars_inside_a_block_stack(
      {std::unordered_set<std::string>{}}
  );

  while (std::cin >> last_string) {
    if (last_string == "}") {
      for (const std::string& var_name : used_vars_inside_a_block_stack.top()) {
        auto var_history_iterator = var_name_to_value_history.find(var_name);

        bool found = var_history_iterator != var_name_to_value_history.end();

        if (found) {
          var_history_iterator->second.pop();

          if (var_history_iterator->second.empty()) {
            // in this case variable was declared inside a block and we can discard it
            var_name_to_value_history.erase(var_history_iterator->first);
          }
        }
      }

      used_vars_inside_a_block_stack.pop();
      continue;
    }

    if (last_string == "{") {
      // push an empty one
      used_vars_inside_a_block_stack.emplace();

      continue;
    }
```

```
    size_t equals_char_pos = last_string.find('=');

    std::string lvalue_name = last_string.substr(0, equals_char_pos);
    std::string rvalue_string = last_string.substr(equals_char_pos + 1);

    bool rvalue_is_number =
        std::all_of(rvalue_string.begin(), rvalue_string.end(), [](char character) -> bo
          return character == '-' || std::isdigit(character) != 0;
        });

    int rvalue = 0;

    if (rvalue_is_number) {
      rvalue = std::stoi(rvalue_string);
    } else {
      auto iterator = var_name_to_value_history.find(rvalue_string);

      if (iterator != var_name_to_value_history.end()) {
        rvalue = iterator->second.top();
      }
      // if value is not found, the default is zero
      std::cout << rvalue << '\n';
    }

    auto emplace_result = var_name_to_value_history.emplace(lvalue_name, std::stack<int>
    // bool new_variable = emplace_result.second;
    // std::stack variable_history = emplace_result.first->second;

    auto is_var_emplaced = used_vars_inside_a_block_stack.top().emplace(lvalue_name).sec
    if (!is_var_emplaced) {
      // variable overwrite
      emplace_result.first->second.pop();
    }

    // push value to the stack
    emplace_result.first->second.push(rvalue);
  }

  return 0;
}

void close_code_block() {
}
```

# 4 Задача D. Профессор Хаос

## 4.1 Пояснение к примененному алгоритму

Time complexity: O(n) Space complexity: O(1)

## 4.2 Код алгоритма

```cpp
#include <iostream>

int main() {
  int a_input = 0;
  int b_input = 0;
  int c_input = 0;
  int d_input = 0;
  int k_input = 0;
  int yesterday_amount = -1;
  std::cin >> a_input;
  std::cin >> b_input;
  std::cin >> c_input;
  std::cin >> d_input;
  std::cin >> k_input;

  while (k_input > 0) {
    a_input *= b_input;

    if (a_input < c_input) {
      std::cout << "0" << '\n';
      return 0;
    }

    a_input -= c_input;
    a_input = std::min(a_input, d_input);

    if (yesterday_amount == a_input) {
      break;
    }

    yesterday_amount = a_input;
    k_input--;
  }

  std::cout << a_input << '\n';
  return 0;
}
```