Федеральное государственное автономное образовательное учреждение высшего образования
"Национальный исследовательский университет "Высшая школа экономики"

Московский институт электроники и математики им. А.Н. Тихонова НИУ ВШЭ
Департамент компьютерной инженерии

Курс: Алгоритмизация и программирование

Раздел	**Макс. оценка**	**Итоговая оценка**
Работа программы	1	
Тесты	1	
Правильность алгоритма	3	
Ответы на вопросы	2	
Дополнительное задание	3	
Итого	10	

Отчет по лабораторной работе № 10

Студент: Чапайкин Арсений Георгиевич

Группа: БИВ242

Вариант: № 171 (2, 4, 4)

Руководитель: Елисеенко А.М.

Оценка:

Дата сдачи:

МОСКВА 2024

Листинг программы

```cpp
#include <array>
#include <fstream>
#include <iostream>
#include <stdexcept>
#include <string>
#include <type_traits>

class Grade {
    unsigned int value : 4;
    friend std::istream& operator>>(std::istream&, Grade&);

public:
    template <typename T = unsigned int>
    requires std::is_integral_v<T>
    explicit Grade(T v = 0) : value(static_cast<unsigned int>(v)) {
        if (v < 0 || v > 10) {
            throw std::out_of_range("Value must be between 0 and 10");
        }
    }

    template <typename T>
    requires std::is_integral_v<T>
    Grade& operator=(T other) {
        if (other < 0 || other > 10) {
            throw std::out_of_range("Value must be between 0 and 10");
        }
        this->value = other;
        return *this;
    }

    unsigned int get_value() const noexcept {
        return value;
    };

    bool operator<(Grade other) const noexcept {
        return value < other.value;
    }
};

std::istream& operator>>(std::istream& is, Grade& g) {
    int temp;
    is >> temp;

    if (temp >= 0 && temp <= 10) {
        g.value = static_cast<unsigned int>(temp);
    } else {
        throw std::out_of_range("Grade is out of range");
    }
    return is;
}

std::ostream& operator<<(std::ostream& os, Grade g) {
    os << g.get_value();
    return os;
}

class Student {
```

```
58      struct _full_name {
59          std::string first_name;
60          std::string last_name;
61          std::string patronimyc;
62      };
63
64      struct _performance {
65          double gpa = 0;
66          std::array<Grade, 4> exam_results;
67      };
68
69  public:
70      _full_name full_name;
71      std::string group;
72      _performance performance;
73  };
74
75  std::istream& operator>>(std::istream& is, Student& s) {
76      is >> s.full_name.first_name;
77      is >> s.full_name.last_name;
78      is >> s.full_name.patronimyc;
79      is >> s.group;
80      for (int i = 0; i < 4; i++) {
81          is >> s.performance.exam_results[i];
82          s.performance.gpa += s.performance.exam_results[i].get_value();
83      }
84      s.performance.gpa /= 4.0;
85      return is;
86  }
87
88  std::ostream& operator<<(std::ostream& os, const Student& s) {
89      os << s.full_name.first_name << '\n';
90      os << s.full_name.last_name << '\n';
91      os << s.full_name.patronimyc << '\n';
92      os << s.group << '\n';
93      for (int i = 0; i < 4; i++) {
94          os << s.performance.exam_results[i] << ' ';
95      }
96      os << '\n';
97      os << s.performance.gpa;
98      return os;
99  }
100
101 template <typename T>
102 class Queue {
103 protected:
104     struct Node {
105         T data;
106         Node* prev;
107         Node* next;
108
109         explicit Node(const T& data, Node* prev = nullptr,
110                       Node* next = nullptr) noexcept
111             : data(data), prev(prev), next(next) {}
112     };
113
114     Node* head = nullptr;
115     size_t size = 0;
116
117 public:
```

```
118     Queue() = default;
119 
120     ~Queue() { clear(); }
121 
122     T& front() {
123         if (empty()) {
124             throw std::out_of_range("List is empty");
125         }
126         return head->data;
127     }
128 
129     T& back() {
130         if (empty()) {
131             throw std::out_of_range("List is empty");
132         }
133         return head->prev->data;
134     }
135 
136     bool empty() const noexcept {
137         return size == 0;
138     }
139 
140     void push(const T& data) {
141         if (empty()) {
142             head = new Node(data);
143             head->prev = head;
144             head->next = head;
145         } else {
146             Node* newNode = new Node(data, head->prev, head);
147             head->prev->next = newNode;
148             head->prev = newNode;
149         }
150         size++;
151     }
152 
153     void pop() {
154         if (empty()) {
155             throw std::out_of_range("List is empty");
156         }
157 
158         if (size == 1) {
159             delete head;
160             head = nullptr;
161         } else {
162             Node* temp = head;
163             head->prev->next = head->next;
164             head->next->prev = head->prev;
165             head = head->next;
166             delete temp;
167         }
168         size--;
169     }
170 
171     void clear() {
172         while (!empty()) {
173             pop();
174         }
175     }
176 
177     size_t Size() const noexcept {
```

```
178         return size;
179     }
180 };
181
182 int main() {
183     std::string input;
184     std::cin >> input;
185
186     std::fstream is(input);
187
188     Queue<Student> students;
189
190     size_t t;
191     is >> t;
192
193     double min_gpa = 10;
194     while (t--) {
195         Student s;
196         is >> s;
197         min_gpa = std::min(min_gpa, s.performance.gpa);
198         students.push(s);
199     }
200
201     size_t sz = students.Size();
202     for (size_t i = 0; i < sz; i++) {
203         Student current = students.front();
204         students.pop();
205         if (current.performance.gpa != min_gpa) {
206             students.push(current);
207         }
208     }
209
210     while (!students.empty()) {
211         std::cout << students.front() << "\n";
212         students.pop();
213     }
214
215     is.close();
216     return 0;
217 }
```

Распечатка тестов к программе и результатов

Задание 1

Номер	Исходные данные	Результат
1	3 John Doe Ivanovich A-101 3 3 3 3 Jane Smith Petrovna B-202 4 4 4 4 Bob Brown Aleksandrovich C-303 2 3 2 3	John Doe Ivanovich A-101 3 3 3 3 3 Jane Smith Petrovna B-202 4 4 4 4 4
2	2 Alice Gray Dmitrievna D-404 3 3 3 3 Mike Black Sergeevich E-505 3 3 3 3	Resulting queue is empty
3	3 Tom White Olegovich F-606 2 2 2 2 Lily Green Nikolaevna G-707 2 2 2 2 Eva Red Artemovna H-808 3 3 3 3	Eva Red Artemovna H-808 3 3 3 3 3