

Министерство науки и высшего образования Российской Федерации  
Федеральное государственное автономное образовательное учреждение  
высшего образования

**«Пермский национальный исследовательский  
политехнический университет»**

Электротехнический факультет  
Кафедра «Информационные технологии и автоматизированные системы»  
направление подготовки: 09.03.01 – «Информатика и вычислительная  
техника»

**Отчет по  
лабораторной работе 13 (классы)**

Выполнил студент гр. ИВТ-23-16  
Давыдов Андрей Юрьевич

Проверил:

ст. преп. каф. ИТАС

Яруллин Денис Владимирович

\_\_\_\_\_  
(оценка)

\_\_\_\_\_  
(подпись)

\_\_\_\_\_  
(дата)

г. Пермь, 2023

## Постановка задачи

### Задача 1

1. Контейнер - список
2. Тип элементов Pair (см. лабораторную работу №3).

### Задача 2

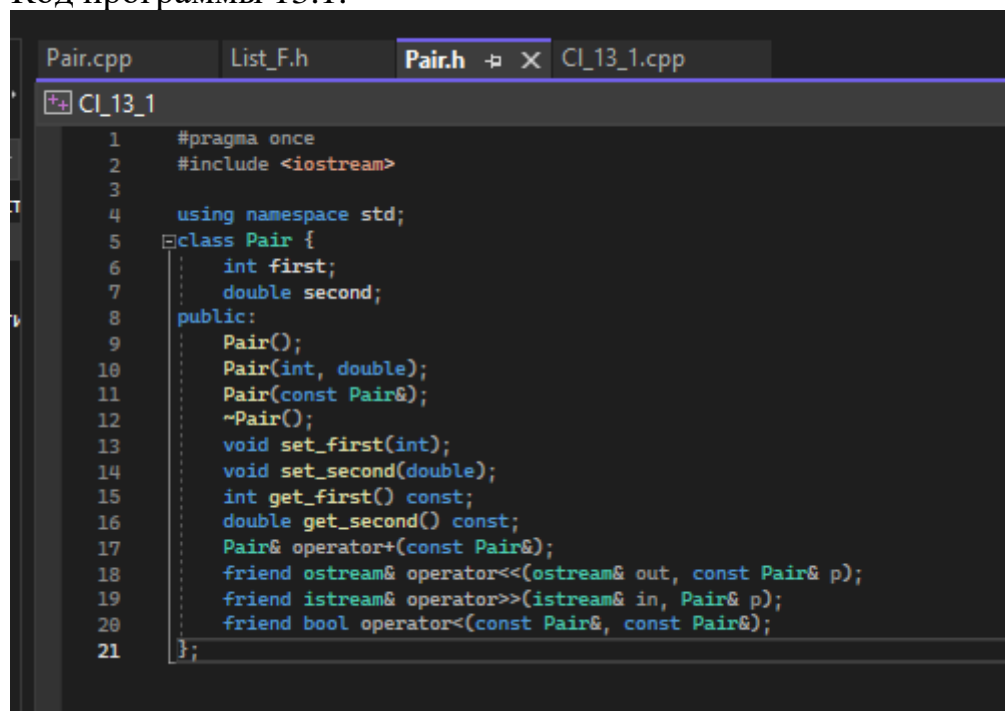
Адаптер контейнера – очередь с приоритетами.

### Задача 3

Ассоциативный контейнер - словарь

Задание 3	Задание 4	Задание 5
Найти среднее арифметическое и добавить его в конец контейнера	Найти элементы ключами из заданного диапазона и удалить их из контейнера	К каждому элементу добавить сумму минимального и максимального элементов контейнера.

Код программы 13.1:



```
Pair.cpp  List_F.h  Pair.h  CI_13_1.cpp
CI_13_1
1  #pragma once
2  #include <iostream>
3
4  using namespace std;
5  class Pair {
6      int first;
7      double second;
8  public:
9      Pair();
10     Pair(int, double);
11     Pair(const Pair&);
12     ~Pair();
13     void set_first(int);
14     void set_second(double);
15     int get_first() const;
16     double get_second() const;
17     Pair& operator+(const Pair&);
18     friend ostream& operator<<(ostream& out, const Pair& p);
19     friend istream& operator>>(istream& in, Pair& p);
20     friend bool operator<(const Pair&, const Pair&);
21 }
```

```
1  #include "Pair.h"
2
3  Pair::Pair() {
4      first = 0;
5      second = 0.0;
6  }
7  Pair::Pair(int f, double s) {
8      first = f;
9      second = s;
10 }
11 Pair::Pair(const Pair& p) {
12     first = p.first;
13     second = p.second;
14 }
15 Pair::~Pair() {}
16 void Pair::set_first(int f) {
17     first = f;
18 }
19 void Pair::set_second(double s) {
20     second = s;
21 }
22 int Pair::get_first() const {
23     return first;
24 }
25 double Pair::get_second() const {
26     return second;
27 }
28 Pair& Pair::operator+(const Pair& p) {
29     first += p.first;
30     second += p.second;
31     return *this;
32 }
33 ostream& operator<<(ostream& out, const Pair& p) {
34     return (out << p.first << " : " << p.second);
35 }
36 istream& operator>>(istream& in, Pair& p) {
37     in >> p.first;
38     in >> p.second;
39     return in;
40 }
41
42 bool operator < (const Pair& pair1, const Pair& pair) {
43     return (pair1.get_first() + pair1.get_second() < pair.get_first() + pair.get_second());
44 }
```

```
Pair.cpp  List_F.h  Pair.h  Cl_13_1.cpp
Cl_13_1 (Глобальная область)
1  #pragma once
2  #include <list>
3  #include <numeric>
4  #include <algorithm>
5  #include "Pair.h"
6
7
8  void Show_list(const list<Pair>& m_list) {
9      if (m_list.size() > 0) {
10         for_each(m_list.begin(), m_list.end(), [](Pair p) {
11             cout << p << endl;
12         });
13     }
14     else {
15         cout << "Список пуст!!";
16     }
17     cout << '\n';
18 }
19
20 void Push_average_value(list<Pair>& m_list) {
21     if (m_list.size() > 0) {
22         Pair Average = accumulate(m_list.begin(), m_list.end(), Pair(0, 0));
23         Average.set_first(Average.get_first() / m_list.size());
24         Average.set_second(Average.get_second() / m_list.size());
25         m_list.push_back(Average);
26     }
27     else {
28         cout << "Список пуст!!\n";
29     }
30 }
31
32 void Delete_range_keys(list<Pair>& m_list, const double Begin_value, const double End_value) {
33     auto i = remove_if(m_list.begin(), m_list.end(), [Begin_value, End_value](Pair p) {
34         return ((Begin_value <= p.get_first() && p.get_first() <= End_value) && (Begin_value <= p.get_second() && p.get_second() <= End_value));
35     });
36     m_list.erase(i, m_list.end());
37 }
38
39 void Plus_sum_Min_Max(list<Pair>& m_list) {
40     if (m_list.size() > 0) {
41         Pair max = *max_element(m_list.begin(), m_list.end());
42         Pair min = *min_element(m_list.begin(), m_list.end());
43         for (Pair& c : m_list) {
44             c + max + min;
45         }
46     }
47 }
```

```
Pair.cpp  List_F.h  Pair.h  Cl_13_1.cpp
Cl_13_1 (Глобальная область)
1  #include "List_F.h"
2  #include "Pair.h"
3
4
5
6  int main() {
7      system("chcp 1251 > null");
8
9
10     cout << "Исходный список" << endl;
11     list<Pair> a = { Pair(6, 2.9), Pair(7, 9), Pair(2, 3.5) };
12     Show_list(a);
13
14     cout << "Список после добавления среднего значения в конец" << endl;
15     Push_average_value(a);
16     Show_list(a);
17
18     double begin, end;
19
20
21     cout << "Введите диапазон, элементы содержащие ключи внутри которого будут удалены\n";
22     cout << "От: ";
23     cin >> begin;
24     cout << "До: ";
25     cin >> end;
26     cout << "Список после удаления элементов содержащих ключи в заданном диапазоне\n";
27     Delete_range_keys(a, begin, end);
28     Show_list(a);
29
30     cout << "Список, после добавления к каждому элементу суммы максимального и минимального элементов\n";
31     Plus_sum_Min_Max(a);
32     Show_list(a);
33
34     return 0;
35 }
```

## Код программы 13.2:

```
Pair.cpp  Que_F.h  Pair.h  Cl_13_2.cpp
Cl_13_2 (Глобальная область)
1  #include "Pair.h"
2
3  Pair::Pair() {
4      first = 0;
5      second = 0.0;
6  }
7  Pair::Pair(int f, double s) {
8      first = f;
9      second = s;
10 }
11 Pair::Pair(const Pair& p) {
12     first = p.first;
13     second = p.second;
14 }
15 Pair::~Pair() {}
16 void Pair::set_first(int f) {
17     first = f;
18 }
19 void Pair::set_second(double s) {
20     second = s;
21 }
22 int Pair::get_first() const {
23     return first;
24 }
25 double Pair::get_second() const {
26     return second;
27 }
28 Pair& Pair::operator+(const Pair& p) {
29     first += p.first;
30     second += p.second;
31     return *this;
32 }
33 ostream& operator<<(ostream& out, const Pair& p) {
34     return (out << p.first << " : " << p.second);
35 }
36 istream& operator>>(istream& in, Pair& p) {
37     in >> p.first;
38     in >> p.second;
39     return in;
40 }
41
42 bool operator < (const Pair& pair1, const Pair& pair) {
43     return (pair1.get_first() + pair1.get_second() < pair.get_first() + pair.get_second());
44 }
```

```
Pair.cpp  Que_F.h  Pair.h  Cl_13_2.cpp
Cl_13_2
1  #pragma once
2  #include <iostream>
3
4  using namespace std;
5  class Pair {
6      int first;
7      double second;
8  public:
9      Pair();
10     Pair(int, double);
11     Pair(const Pair&);
12     ~Pair();
13     void set_first(int);
14     void set_second(double);
15     int get_first() const;
16     double get_second() const;
17     Pair& operator+(const Pair&);
18     friend ostream& operator<<(ostream& out, const Pair& p);
19     friend istream& operator>>(istream& in, Pair& p);
20     friend bool operator<(const Pair&, const Pair&);
21 }
```

```

Pair.cpp  Que_F.h  Pair.h  Cl_13_2.cpp
Cl_13_2  (Глобальная область)

1  #pragma once
2  #include <iostream>
3  #include <list>
4  #include <numeric>
5  #include <algorithm>
6  #include <Queue>
7  #include "Pair.h"
8
9  using namespace std;
10
11  list<Pair> priority_queueTolist(priority_queue<Pair>& m_priority_queue) {
12      priority_queue<Pair> tmp_priority_queue = m_priority_queue;
13      list<Pair> m_list;
14      while (tmp_priority_queue.size() > 0) {
15          m_list.push_back(tmp_priority_queue.top());
16          tmp_priority_queue.pop();
17      }
18      return m_list;
19  }
20
21  void Push_average_value(priority_queue<Pair>& m_priority_queue) {
22      if (m_priority_queue.size() > 0) {
23          list<Pair> m_list = priority_queueTolist(m_priority_queue);
24          Pair Average = accumulate(m_list.begin(), m_list.end(), Pair(0, 0));
25          Average.set_first(Average.get_first() / m_list.size());
26          Average.set_second(Average.get_second() / m_list.size());
27          m_priority_queue.push(Average);
28      }
29      else {
30          cout << "Очередь пуста!\n";
31      }
32  }
33
34  void Delete_range_keys(priority_queue<Pair>& m_priority_queue, const double BeginValue, const double EndValue) {
35      if (m_priority_queue.size() > 0) {
36          list<Pair> m_list = priority_queueTolist(m_priority_queue);
37          auto i = remove_if(m_list.begin(), m_list.end(), [BeginValue, EndValue](Pair p) {
38              return (BeginValue <= p.get_first() && p.get_first() <= EndValue) && (BeginValue <= p.get_second() && p.get_second() <= EndValue);
39          });
40          m_priority_queue = {};
41          if (i != m_list.begin()) {
42              for_each(m_list.begin(), i--, [&m_priority_queue](Pair p) {
43                  m_priority_queue.push(p);
44              });
45          }
46      }
47  }
48
49  void Plus_sum_Min_Max(priority_queue<Pair>& m_priority_queue) {
50      if (m_priority_queue.size() > 0) {
51          list<Pair> m_list = priority_queueTolist(m_priority_queue);
52          m_priority_queue = {};
53          Pair max = *max_element(m_list.begin(), m_list.end());
54          Pair min = *min_element(m_list.begin(), m_list.end());
55          for (Pair& c : m_list) {
56              m_priority_queue.push(c + max + min);
57          }
58      }
59  }
60
61  void Show_priority_queue(priority_queue<Pair>& m_priority_queue) {
62      if (m_priority_queue.size() > 0) {
63          list<Pair> m_list = priority_queueTolist(m_priority_queue);
64          for_each(m_list.begin(), m_list.end(), [](Pair p) {
65              cout << p << '\n';
66          });
67      }
68      else {
69          cout << "Очередь пуста!";
70      }
71      cout << '\n';
72  }

```

```
Pair.cpp  Que_F.h  Pair.h  Cl_13_2.cpp  X
Cl_13_2  (Глобальная область)
6  int main() {
7      system("chcp 1251 > null");
8
9      priority_queue <Pair> a;
10     for (Pair i : { Pair(4, 2.52), Pair(5, 2.5), Pair(10, 23.5) }) {
11         a.push(i);
12     }
13     cout << "Исходная очередь с приоритетом" << endl;
14     Show_priority_queue(a);
15
16     cout << "Очередь с приоритетом после добавления среднего значения в конец" << endl;
17     Push_average_value(a);
18     Show_priority_queue(a);
19
20     double begin, end;
21
22     cout << "Введите диапазон, элементы содержащие ключи внутри которого будут удалены\n";
23     cout << "От: ";
24     cin >> begin;
25     cout << "До: ";
26     cin >> end;
27     cout << "Очередь с приоритетом, после удаления элементов содержащих ключи в заданном диапазоне\n";
28     Delete_range_keys(a, begin, end);
29     Show_priority_queue(a);
30
31     cout << "Очередь с приоритетом, после добавления к каждому элементу суммы максимального и минимального элементов\n";
32     Plus_sum_Min_Max(a);
33     Show_priority_queue(a);
34
35     return 0;
36 }
```

Код программы 13.3

```
Pair.cpp  X  Map_F.h  Pair.h  X  Cl_13_3.cpp
Cl_13_3
1  #pragma once
2  #include <iostream>
3
4  using namespace std;
5  class Pair {
6      int first;
7      double second;
8  public:
9      Pair();
10     Pair(int, double);
11     Pair(const Pair&);
12     ~Pair();
13     void set_first(int);
14     void set_second(double);
15     int get_first() const;
16     double get_second() const;
17     Pair& operator+(const Pair&);
18     Pair& operator/(int);
19     friend ostream& operator<<(ostream& out, const Pair& p);
20     friend istream& operator>>(istream& in, Pair& p);
21     friend bool operator <=(const Pair&, double s);
22     friend bool operator <= (double, const Pair&);
23     friend bool operator <(const Pair&, const Pair&);
24 }
```

```
1  #include "Pair.h"
2
3  Pair::Pair() {
4      first = 0;
5      second = 0.0;
6  }
7  Pair::Pair(int f, double s) {
8      first = f;
9      second = s;
10 }
11 Pair::Pair(const Pair& p) {
12     first = p.first;
13     second = p.second;
14 }
15 Pair::~Pair() {}
16 void Pair::set_first(int f) {
17     first = f;
18 }
19 void Pair::set_second(double s) {
20     second = s;
21 }
22 int Pair::get_first() const {
23     return first;
24 }
25 double Pair::get_second() const {
26     return second;
27 }
28 Pair& Pair::operator+(const Pair& p) {
29     first += p.first;
30     second += p.second;
31     return *this;
32 }
33 Pair& Pair::operator/(int d) {
34     first /= d;
35     second /= d;
36     return *this;
37 }
38 ostream& operator<<(ostream& out, const Pair& p) {
39     return (out << p.first << " : " << p.second);
40 }
41 istream& operator>>(istream& in, Pair& p) {
42     in >> p.first;
43     in >> p.second;
44     return in;
45 }
46
47 bool operator <= (const Pair& pair1, double s) {
48     return (pair1.get_first() <= s and pair1.get_second() <= s);
49 }
50 bool operator <= (double s, const Pair& pair1) {
51     return (pair1.get_first() >= s and pair1.get_second() >= s);
52 }
53 bool operator < (const Pair& pair1, const Pair& pair) {
54     return (pair1.get_first() + pair1.get_second() < pair.get_first() + pair.get_second());
55 }
```



```
1  #pragma once
2  #include <map>
3  #include <iostream>
4  #include <list>
5  #include <numeric>
6  #include <algorithm>
7
8
9  using namespace std;
10
11 void Push_average_value(map <int, Pair>& m_map) {
12     if (m_map.size() > 0) {
13         int MaxKey = max_element(m_map.begin(), m_map.end(), [](const auto& p1, const auto& p2) {
14             return p1.first < p2.first;
15         })->first + 1;
16         Pair Middle_second = accumulate(m_map.begin(), m_map.end(), Pair(0, 0), [](Pair value, const auto& p) {
17             return value + p.second;
18         }) / m_map.size();
19         m_map[MaxKey] = Middle_second;
20         m_map.insert({ MaxKey, Middle_second });
21     }
22     else {
23         cout << "Словарь пуст!\n";
24     }
25 }
26
27 void Delete_range_keys(map <int, Pair>& m_map, const double BeginValue, const double EndValue) {
28     for (auto p = m_map.begin(); p != m_map.end(); ) {
29         if (BeginValue <= (*p).second && (*p).second <= EndValue) {
30             p = m_map.erase(p);
31         }
32         else {
33             p++;
34         }
35     };
36 }
37
38 void Plus_sum_Min_Max(map <int, Pair>& m_map) {
39     if (m_map.size() > 0) {
40         Pair MaxValue = max_element(m_map.begin(), m_map.end(), [](const auto& p1, const auto& p2) {
41             return p1.second < p2.second;
42         })->second;
43         Pair MinValue = min_element(m_map.begin(), m_map.end(), [](const auto& p1, const auto& p2) {
44             return p1.second < p2.second;
45         })->second;
46         for (auto& p : m_map) {
47             p.second = p.second + MaxValue + MinValue;
48         }
49     }
50 }
51
52 void Show_map(const map <int, Pair>& m_map) {
53     if (m_map.size() > 0) {
54         for (auto& p : m_map) {
55             cout << p.first << " " << p.second << '\n';
56         }
57     }
58     else {
59         cout << "Словарь пуст!";
60     }
61     cout << '\n';
62 }
```

```
Pair.cpp  Map_F.h  Pair.h  Cl_13_3.cpp  (Глобальная область)

1  #include "Map_F.h"
2  #include "Pair.h"
3
4
5  int main() {
6      system("chcp 1251 > null*");
7
8      map<int, Pair> a = { {1, {2, 3.14}}, {2, {4, 5.44}}, {3, {12, 2.15}}, {4, {2, 6.28}} };
9      cout << "Исходный словарь" << endl;
10     Show_map(a);
11
12     cout << "Словарь после добавления среднего значения в конец" << endl;
13     Push_average_value(a);
14     Show_map(a);
15
16     double begin, end;
17     cout << "Введите диапазон, элементы содержащие ключи, внутри которого будут удалены\n";
18     cin >> begin;
19     cin >> end;
20     Delete_range_keys(a, begin, end);
21     cout << "Слова, после удаления элементов содержащих ключи в заданном диапазоне\n";
22     Show_map(a);
23
24     cout << "Словарь, после добавления к каждому элементу суммы максимального и минимального элементов\n";
25     Plus_sum_Min_Max(a);
26     Show_map(a);
27
28     return 0;
29 }
```