Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное

учреждение

высшего образования

«**Пермский национальный исследовательский**

**политехнический университет»**

Кафедра «Информационные технологии и автоматизированные

системы»

**ОТЧЁТ**

**по лабораторной работе №18.13**

Дисциплина: «Информатика»

Тема: “Объектно-ориентированное программирование.

Стандартные обобщенные алгоритмы библиотеки STL.”

Вариант 13

Выполнил:

Студент группы ИВТ-20-2б

Сафронов Владислав Владиславович

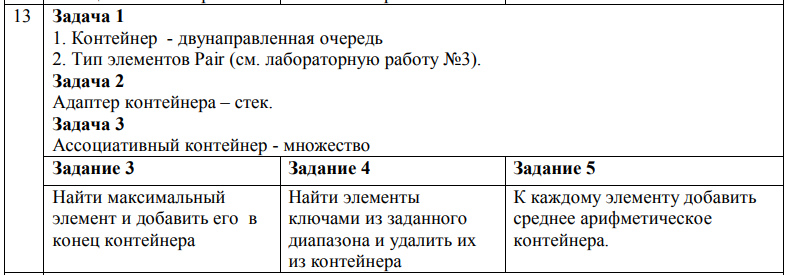
Проверила:

Доцент кафедры ИТАС

Полякова О.А.

**Пермь, 2021**

**Постановка задачи**

****

**Анализ задачи**

1. Для решения задачи необходимо:

1.1. Использовать класс Pair.

1.2. Использовать контейнеры библиотеки STL.

1.3. Использовать алгоритмы библиотеки STL.

1. В ходе работы были использованы следующие типы данных:
   1. Pair, int, double
2. Для решения задачи были использованы следующие типы данных:
   1. deque<Pair> deq;
   2. stack<Pair> st;
   3. set<Pair> pset;
3. Для ввода и вывода использовались следующие операторы и функции:
   1. Операторы << и >>.
4. Поставленные задачи будут решены следующими действиями:
   1. Функция для удаления элементов из контейнера в диапазоне:
5. stack<Pair> delete\_task(stack<Pair> a, Pair left, Pair right)
6. {
7. stack<Pair> tmp, tmp\_2;
8. int count = 0;
9. for (int i = 0; i < a.size(); i++)
10. {
11. if (a.top() < left || a.top() > right)
12. {
13. tmp.push(a.top());
14. count++;
15. }
16. a.pop();
17. }
18. for (int i = 0; i < count; i++)
19. {
20. tmp\_2.push(tmp.top());
21. tmp.pop();
22. }
23. return tmp\_2;
24. }
    1. Функция для добавления среднего арифметического к каждому элементу контейнера:
25. set<Pair> add\_average(set<Pair> m)
26. {
27. set<Pair>::iterator iter = m.begin();
28. set<Pair> temp\_set;
29. int size = 0;
30. Pair avg;
31. while (iter != m.end())
32. {
33. avg += \*iter;
34. size++;
35. iter++;
36. }
37. avg /= size;
38. iter = m.begin();
39. while (iter != m.end())
40. {
41. Pair tmp = avg + (\*iter);
42. iter++;
43. temp\_set.insert(tmp);
44. }
45. return temp\_set;
46. }

**Блок-схема программы**

**Код программы**

**Файл Pair.h:**

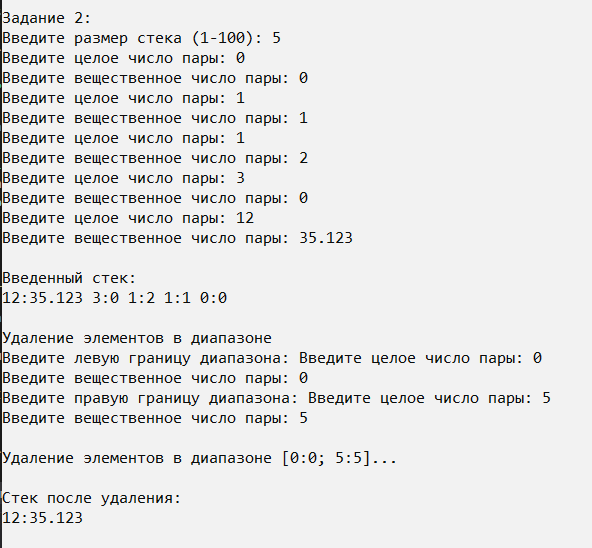
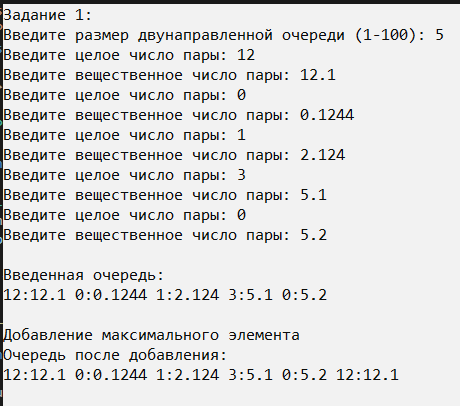
1. #pragma once
2. #include <iostream>
3. using namespace std;
5. class Pair
6. {
7. int first;
8. double second;
9. public:
10. Pair()
11. {
12. first = 0;
13. second = 0;
14. }
16. Pair(int f, double s)
17. {
18. this->first = f;
19. this->second = s;
20. }
21. Pair(const Pair& obj)
22. {
23. first = obj.first;
24. second = obj.second;
25. }
27. void Show()
28. {
29. cout << first << ":" << second << endl;
30. }
31. void SetFirst(int n)
32. {
33. first = n;
34. }
35. void SetSecond(double n)
36. {
37. second = n;
38. }
39. int GetFirst()
40. {
41. return first;
42. }
43. double GetSecond()
44. {
45. return second;
46. }
48. friend bool operator>(const Pair& obj1, const Pair& obj2);
49. friend bool operator>=(const Pair& obj1, const Pair& obj2);
50. friend bool operator<(const Pair& obj1, const Pair& obj2);
51. friend bool operator<=(const Pair& obj1, const Pair& obj2);
52. friend bool operator==(const Pair& obj1, const Pair& obj2);
53. friend bool operator!=(const Pair& obj1, const Pair& obj2);
55. Pair& operator--() // префикс
56. {
57. first--;
58. return \*this;
59. }
60. Pair operator--(int) //постфикс
61. {
62. second--;
63. return \*this;
64. }
65. Pair operator + (int k)
66. {
67. return Pair(first + k, second);
68. }
69. Pair operator + (double k)
70. {
71. return Pair(first, second + k);
72. }
73. Pair operator + (const Pair& a)
74. {
75. return Pair(first + a.first, second + a.second);
76. }
77. Pair operator - (int k)
78. {
79. return Pair(first - k, second);
80. }
81. Pair operator - (double k)
82. {
83. return Pair(first, second - k);
84. }
85. Pair operator - (Pair& a)
86. {
87. return Pair(first - a.first, second - a.second);
88. }
89. Pair operator / (int k)
90. {
91. return Pair(first / k, second);
92. }
93. Pair operator / (double k)
94. {
95. return Pair(first, second / k);
96. }
97. Pair operator / (Pair& a)
98. {
99. return Pair(first / a.first, second / a.second);
100. }
102. void operator += (int k)
103. {
104. first += k;
105. }
106. void operator += (double k)
107. {
108. second += k;
109. }
110. void operator += (const Pair& a)
111. {
112. first += a.first;
113. second += a.second;
114. }
116. void operator -= (int k)
117. {
118. first -= k;
119. }
120. void operator -= (double k)
121. {
122. second -= k;
123. }
124. void operator -= (const Pair& a)
125. {
126. first -= a.first;
127. second -= a.second;
128. }
130. void operator /= (int k)
131. {
132. first /= k; second /= k;
133. }
134. void operator /= (Pair& a)
135. {
136. first /= a.first;
137. second /= a.second;
138. }
140. friend ostream& operator << (ostream& out, const Pair& a)
141. {
142. out << a.first << ":" << a.second;
143. return out;
144. }
146. friend istream& operator >> (istream& in, Pair& a)
147. {
148. cout << "¬ведите целое число пары: "; in >> a.first;
149. cout << "¬ведите вещественное число пары: "; in >> a.second;
150. return in;
151. }
153. ~Pair() {};
154. };
156. bool operator>(const Pair& obj1, const Pair& obj2)
157. {
158. //return (obj1.first > obj2.first && obj1.second > obj2.second);
159. return (((double)obj1.first + obj1.second) > ((double)obj2.first + obj2.second));
160. }
162. bool operator>=(const Pair& obj1, const Pair& obj2)
163. {
164. //return (obj1.first >= obj2.first && obj1.second >= obj2.second);
165. return (((double)obj1.first + obj1.second) >= ((double)obj2.first + obj2.second));
166. }
168. bool operator<(const Pair& obj1, const Pair& obj2)
169. {
170. //return (obj1.first < obj2.first && obj1.second < obj2.second);
171. return (((double)obj1.first + obj1.second) < ((double)obj2.first + obj2.second));
172. }
174. bool operator<=(const Pair& obj1, const Pair& obj2)
175. {
176. //return (obj1.first <= obj2.first && obj1.second <= obj2.second);
177. return (((double)obj1.first + obj1.second) <= ((double)obj2.first + obj2.second));
178. }
180. bool operator==(const Pair& obj1, const Pair& obj2)
181. {
182. return (obj1.first == obj2.first && obj1.second == obj2.second);
183. }
185. bool operator!=(const Pair& obj1, const Pair& obj2)
186. {
187. return (obj1.first != obj2.first || obj1.second != obj2.second);
188. }

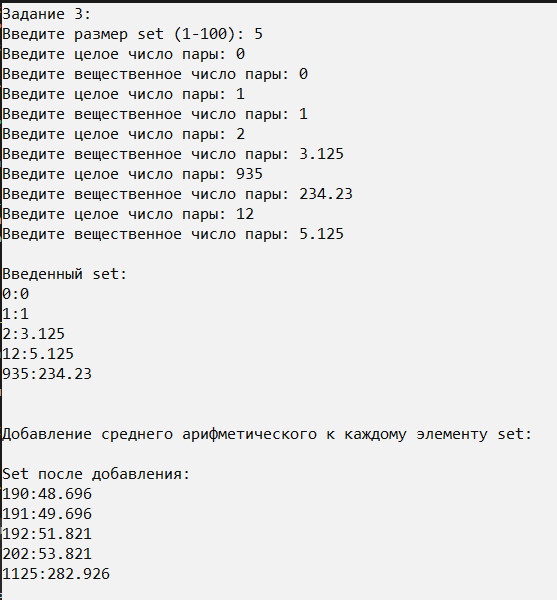
**Файл main.cpp:**

1. #include <deque>
2. #include <iostream>
3. #include <algorithm>
4. #include <stack>
5. #include <set>
6. #include "Pair.h"
7. using namespace std;
9. deque<Pair> make\_deque(int size)
10. {
11. deque<Pair> tmp;
13. for (int i = 0; i < size; i++)
14. {
15. Pair tmp\_pair;
16. cin >> tmp\_pair;
17. tmp.push\_back(tmp\_pair);
18. }
19. return tmp;
20. }
21. void print\_deque(deque<Pair> tmp)
22. {
23. for (int i = 0; i < tmp.size(); i++)
24. {
25. cout << tmp[i] << " ";
26. }
27. cout << endl;
28. }
30. stack<Pair> make\_stack(int size)
31. {
32. stack<Pair> tmp;
34. for (int i = 0; i < size; i++)
35. {
36. Pair tmp\_pair;
37. cin >> tmp\_pair;
38. tmp.push(tmp\_pair);
39. }
40. return tmp;
41. }
42. void print\_stack(stack<Pair> a)
43. {
44. stack<Pair> tmp = a;
45. while (!tmp.empty())
46. {
47. cout << tmp.top() << " ";
48. tmp.pop();
49. }
50. cout << endl;
51. }
52. stack<Pair> delete\_task(stack<Pair> a, Pair left, Pair right)
53. {
54. stack<Pair> tmp, tmp\_2;
55. int count = 0;
56. for (int i = 0; i < a.size(); i++)
57. {
58. if (a.top() < left || a.top() > right)
59. {
60. tmp.push(a.top());
61. count++;
62. }
63. a.pop();
64. }
65. for (int i = 0; i < count; i++)
66. {
67. tmp\_2.push(tmp.top());
68. tmp.pop();
69. }
70. return tmp\_2;
71. }
73. set<Pair> make\_set(int size)
74. {
75. set<Pair> tmp;
76. for (int i = 0; i < size; i++)
77. {
78. Pair tmp\_pair;
79. cin >> tmp\_pair;
80. tmp.insert(tmp\_pair);
81. }
82. return tmp;
83. }
84. void print\_set(set<Pair> m)
85. {
86. set<Pair>::iterator iter = m.begin();
87. for (; iter != m.end(); iter++)
88. {
89. cout << \*iter << endl;
90. }
91. cout << endl;
92. }
93. set<Pair> add\_average(set<Pair> m)
94. {
95. set<Pair>::iterator iter = m.begin();
96. set<Pair> temp\_set;
97. int size = 0;
98. Pair avg;
99. while (iter != m.end())
100. {
101. avg += \*iter;
102. size++;
103. iter++;
104. }
105. avg /= size;
106. iter = m.begin();
107. while (iter != m.end())
108. {
109. Pair tmp = avg + (\*iter);
110. iter++;
111. temp\_set.insert(tmp);
112. }
113. return temp\_set;
114. }

117. void task1()
118. {
119. deque<Pair> deq;
120. int size;
121. cout << "Введите размер двунаправленной очереди (1-100): "; cin >> size;
122. while (size < 1 || size > 100)
123. {
124. cout << "Ввод некорректен!\n";
125. cout << "Введите размер двунаправленной очереди (1-100): "; cin >> size;
126. }
127. deq = make\_deque(size);
128. cout << "\nВведенная очередь: \n";
129. print\_deque(deq);
130. cout << "\nДобавление максимального элемента";
131. deque<Pair>::iterator max;
132. //deque<Pair>::iterator iter = deq.end();
133. max = max\_element(deq.begin(), deq.end());
134. deq.insert(deq.end(), \*max);
135. cout << "\nОчередь после добавления: \n";
136. print\_deque(deq);
137. }
139. void task2()
140. {
141. stack<Pair> st;
142. int size;
143. Pair left, right;
144. cout << "Введите размер стека (1-100): "; cin >> size;
145. while (size < 1 || size > 100)
146. {
147. cout << "Ввод некорректен!\n";
148. cout << "Введите размер стека (1-100): "; cin >> size;
149. }
150. st = make\_stack(size);
151. cout << "\nВведенный стек:\n";
152. print\_stack(st);
154. cout << "\nУдаление элементов в диапазоне";
155. cout << "\nВведите левую границу диапазона: "; cin >> left;
156. cout << "Введите правую границу диапазона: "; cin >> right;
157. cout << "\nУдаление элементов в диапазоне [" << left << "; " << right << "]...\n";
158. st = delete\_task(st, left, right);
159. cout << "\nСтек после удаления:\n";
160. print\_stack(st);
161. }
163. void task3()
164. {
165. set<Pair> pset;
166. int size;
167. cout << "Введите размер set (1-100): "; cin >> size;
168. while (size < 1 || size > 100)
169. {
170. cout << "Ввод некорректен!\n";
171. cout << "Введите размер set (1-100): "; cin >> size;
172. }
173. pset = make\_set(size);
174. cout << "\nВведенный set:\n";
175. print\_set(pset);
176. cout << "\nДобавление среднего арифметического к каждому элементу set:\n";
177. pset = add\_average(pset);
178. cout << "\nSet после добавления: \n";
179. print\_set(pset);
180. }
182. int main()
183. {
184. setlocale(LC\_ALL, "Russian");
185. cout << "Задание 1:\n";
186. task1();
187. cout << "\nЗадание 2:\n";
188. task2();
189. cout << "\nЗадание 3:\n";
190. task3();
191. return 0;
192. }

**Скриншоты результатов работы программы**

****

****