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 №18.11**

Дисциплина: «Информатика»

Тема: “Объектно-ориентированное программирование.

Последовательные контейнеры библиотеки STL.”

Вариант 13

Выполнил:

Студент группы ИВТ-20-2б

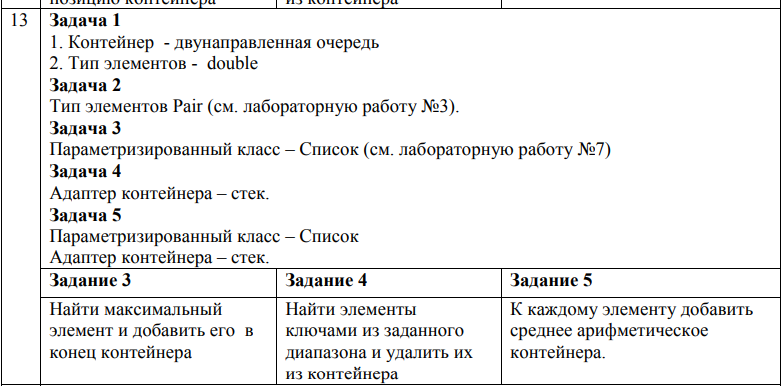
Сафронов Владислав Владиславович

Проверила:

Доцент кафедры ИТАС

Полякова О.А.

**Пермь, 2021**

**Постановка задачи**

**Анализ задачи**

1. Для решения задачи необходимо:
   1. Использовать класс Pair.
   2. Использовать контейнеры библиотеки STL.
2. В ходе работы были использованы следующие типы данных:
   1. Pair, double, int.
3. Для решения задачи данные были представлены в следующем виде:
4. deque<double> temp;
5. deque<Pair> deq;
6. List<double> lst(size, 0);
7. stack<double> st;
8. stack<double> tmp;
9. Для операций ввода и вывода использовались следующие операторы и функции:
   1. Операторы >> и <<.
10. Поставленные задачи будут решены следующими действиями:
    1. Функции для нахождения максимального элемента и добавления его в конец контейнера:
11. template<typename T>
12. deque<T> Add\_Max\_Deque(deque<T> a)
13. {
14. deque<T> tmp = a;
15. T max = tmp[0];
16. for (int i = 0; i < tmp.size(); i++)
17. {
18. if (tmp[i] > max) max = tmp[i];
19. }
20. tmp.push\_back(max);
21. return tmp;
22. }
    1. Функции для удаления из контейнера чисел из заданного диапазона:
23. template <typename T>
24. deque<T> Delete\_Deque(deque<T> a, T left, T right)
25. {
26. deque<T> temp;
27. for (int i = 0; i < a.size(); i++)
28. {
29. if (a[i] < left || a[i] > right) temp.push\_back(a[i]);
30. }
31. return temp;
32. }
    1. Функция для добавления среднего арифметического к каждому элементу контейнера:
33. template <typename T>
34. deque<T> Add\_Middle\_Deque(deque<T> a)
35. {
36. deque<T> tmp = a;
37. T middle = a[0];
38. for (int i = 1; i < tmp.size(); i++) middle += tmp[i];
39. middle /= tmp.size();
40. for (int i = 0; i < tmp.size(); i++) tmp[i] += middle;
41. return tmp;
42. }

**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 + (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 += (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 -= (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, 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 List.h:**

1. #pragma once
2. #include <iostream>
3. using namespace std;
5. template <class T>
6. class List
7. {
8. int size;
9. T\* data;
10. public:
11. List(int s, T k);
12. List(const List<T>& a);
13. ~List();
15. List& operator = (const List<T>& a);
16. T& operator [] (int index);
17. List operator + (const T k);
18. List operator + (const List<T>& a);
19. void push\_back(T a);
20. void erase(int pos);
21. int operator () ();
22. friend ostream& operator << <>(ostream& out, const List<T>& a);
23. friend istream& operator >> <>(istream& in, List<T>& a);
24. };
26. template <class T>
27. List<T>::List(int s, T k)
28. {
29. size = s;
30. data = new T[size];
31. for (int i = 0; i < size; i++) data[i] = k;
32. }
33. template <class T>
34. List<T>::List(const List& a)
35. {
36. size = a.size;
37. data = new T[size];
38. for (int i = 0; i < size; i++) data[i] = a.data[i];
39. }
40. template <class T>
41. List<T>::~List()
42. {
43. delete[] data;
44. data = 0;
45. }
46. template <class T>
47. void List<T>::push\_back(T a)
48. {
49. T\* old\_data = data;
50. data = new T[size + 1];
51. for (int i = 0; i < size; i++) data[i] = old\_data[i];
52. data[size] = a;
53. delete[] old\_data;
54. size++;
55. }
56. template <class T>
57. void List<T>::erase(int pos)
58. {
59. T\* old\_data = data;
60. delete data;
61. data = new int[size - 1];
62. for (int i = 0; i < pos; i++) data[i] = old\_data[i];
63. size--;
64. for (int i = pos; i < size; i++) data[i] = old\_data[i + 1];
65. delete[] old\_data;
66. }
67. template<class T>
68. List<T>& List<T>::operator=(const List<T>& a)
69. {
70. if (this == &a) return \*this;
71. size = a.size;
72. if (data != 0) delete[] data;
73. data = new T[size];
74. for (int i = 0; i < size; i++) data[i] = a.data[i];
75. return \*this;
76. }
77. template<class T>
78. T& List<T>::operator[](int index)
79. {
80. if (index < size) return data[index];
81. else cout << "\nError! Index out of range.";
82. }
83. template<class T>
84. List<T> List<T>::operator+(const T k)
85. {
86. List<T> temp(size, k);
87. for (int i = 0; i < size; ++i) temp.data[i] = data[i] + k;
88. return temp;
89. }
90. template<class T>
91. List<T> List<T>::operator+(const List<T>& a)
92. {
93. int min = size < a.size ? size : a.size;
94. List<T> temp(min, 0);
95. for (int i = 0; i < min; i++) temp.data[i] = data[i] + a.data[i];
96. return temp;
97. }
98. template<class T>
99. int List<T>::operator () ()
100. {
101. return size;
102. }
103. template<class T>
104. ostream& operator << (ostream& out, const List<T>& a)
105. {
106. for (int i = 0; i < a.size; ++i) out << a.data[i] << " ";
107. return out;
108. }
109. template<class T>
110. istream& operator >> (istream& in, List<T>& a)
111. {
112. for (int i = 0; i < a.size; ++i) in >> a.data[i];
113. return in;
114. }

**Файл main.cpp:**

1. #include <iostream>
2. #include <deque>
3. #include <queue>
4. #include <stack>
5. #include <ctime>
6. #include "Pair.h"
7. #include "List.h"
8. using namespace std;
10. template <typename T>
11. void Print\_Deque(deque<T> tmp)
12. {
13. for (int i = 0; i < tmp.size(); i++) cout << tmp[i] << " ";
14. cout << endl;
15. }
16. template <typename T>
17. deque<T> Delete\_Deque(deque<T> a, T left, T right)
18. {
19. deque<T> temp;
20. for (int i = 0; i < a.size(); i++)
21. {
22. if (a[i] < left || a[i] > right) temp.push\_back(a[i]);
23. }
24. return temp;
25. }
26. //template <typename T>
27. //List<T> Delete\_List(List<T> a, T left, T right)
28. //{
29. // int count = 0;
30. // for (int i = 0; i < a(); i++)
31. // {
32. // if (a[i] > left || a[i] < right) {
33. // a[i] = -1;
34. // count++;
35. // }
36. // }
37. // List<T> temp(size - count, 0);
38. // int i = 0;
39. // for (int j = 0; j < a(); j++)
40. // {
41. // if (a[j] != -1) {
42. // temp[i] = a[j];
43. // i++;
44. // }
45. // }
46. // return temp;
47. //}
48. template <typename T>
49. deque<T> Add\_Middle\_Deque(deque<T> a)
50. {
51. deque<T> tmp = a;
52. T middle = a[0];
53. for (int i = 1; i < tmp.size(); i++) middle += tmp[i];
54. middle /= tmp.size();
55. for (int i = 0; i < tmp.size(); i++) tmp[i] += middle;
56. return tmp;
57. }
58. template<typename T>
59. deque<T> Add\_Max\_Deque(deque<T> a)
60. {
61. deque<T> tmp = a;
62. T max = tmp[0];
63. for (int i = 0; i < tmp.size(); i++)
64. {
65. if (tmp[i] > max) max = tmp[i];
66. }
67. tmp.push\_back(max);
68. return tmp;
69. }
71. template<typename T>
72. void Print\_Stack(stack<T> a)
73. {
74. stack<T> temp = a;
75. while (!temp.empty())
76. {
77. cout << temp.top() << " ";
78. temp.pop();
79. }
80. cout << endl;
81. }
83. void Task1()
84. {
85. deque<double> temp;
86. int size;
87. cout << "Введите размер очереди (1-100): "; cin >> size;
88. while (size < 1 || size > 100)
89. {
90. cout << "Ввод некорректен!\n";
91. cout << "Введите размер очереди (1-100): "; cin >> size;
92. }
93. for (int i = 0; i < size; i++)
94. {
95. temp.push\_back(rand() % 10000 / 100.0);
96. }
97. cout << "Сгенерированная очередь: \n";
98. Print\_Deque(temp);
100. cout << "\nДобавление максимума в конец очереди\n";
101. temp = Add\_Max\_Deque(temp);
102. cout << "Очередь после добавления максимального числа в конец:\n";
103. Print\_Deque(temp);
105. cout << "\nУдаление из очереди чисел в заданном диапазоне\n";
106. double min, max;
107. cout << "Введите левую границу диапазона: "; cin >> min;
108. cout << "Введите правую границу диапазона: "; cin >> max;
109. cout << "Удаление из очереди элементов в диапазоне [" << min << ";" << max << "]\n";
110. temp = Delete\_Deque(temp, min, max);
111. cout << "Очередь после удаления:\n";
112. Print\_Deque(temp);
113. cout << "\nДобавление среднего арифметического к каждому элементу очереди:\n";
114. temp = Add\_Middle\_Deque(temp);
115. cout << "Очередь после добавления:\n";
116. Print\_Deque(temp);
117. }
119. void Task2()
120. {
121. Pair a;
122. int size;
123. deque<Pair> deq;
124. cout << "Введите размер очереди (1-100): "; cin >> size;
125. while (size < 1 || size > 100)
126. {
127. cout << "Ввод некорректен!\n";
128. cout << "Введите размер очереди (1-100): "; cin >> size;
129. }
130. for (int i = 0; i < size; i++)
131. {
132. cout << "Пара " << i + 1 << ":\n";
133. Pair tmp;
134. cin >> tmp;
135. deq.push\_back(tmp);
136. }
137. cout << "Полученная очередь: \n";
138. Print\_Deque(deq);
140. cout << "\nДобавление максимума в конец очереди\n";
141. deq = Add\_Max\_Deque(deq);
142. cout << "Очередь после добавления максимального числа в конец:\n";
143. Print\_Deque(deq);
145. cout << "\nУдаление из очереди чисел в заданном диапазоне\n";
146. Pair min, max;
147. cout << "Введите левую границу диапазона: \n"; cin >> min;
148. cout << "Введите правую границу диапазона: \n"; cin >> max;
149. cout << "Удаление из очереди элементов в диапазоне [" << min << ";" << max << "]\n";
150. deq = Delete\_Deque(deq, min, max);
151. cout << "Очередь после удаления:\n";
152. Print\_Deque(deq);
154. cout << "\nДобавление среднего арифметического к каждому элементу очереди:\n";
155. deq = Add\_Middle\_Deque(deq);
156. cout << "Очередь после добавления:\n";
157. Print\_Deque(deq);
158. }
160. void Task3()
161. {
162. int size;
163. cout << "Введите размер списка (1-100): "; cin >> size;
164. while (size < 1 || size > 100)
165. {
166. cout << "Ввод некорректен!\n";
167. cout << "Введите размер списка (1-100): "; cin >> size;
168. }
170. List<double> lst(size, 0);
171. for (int i = 0; i < size; i++) lst[i] = rand() % 10000 / 100.0;
172. cout << "Сгенерированный список: \n";
173. cout << lst << endl;
175. cout << "\nДобавление максимума в конец списка\n";
176. double max = lst[0];
177. for (int i = 1; i < lst(); i++) if (lst[i] > max) max = lst[i];
178. lst.push\_back(max);
179. cout << "Полученный список: \n";
180. cout << lst << endl;
182. cout << "\nУдаление из списка чисел в заданном диапазоне\n";
183. double min;
184. cout << "Введите левую границу диапазона: \n"; cin >> min;
185. cout << "Введите правую границу диапазона: \n"; cin >> max;
186. cout << "Удаление из очереди элементов в диапазоне [" << min << ";" << max << "]\n";
187. /\*for (int i = 0; i < lst(); i++)
188. {
189. if (lst[i] > min || lst[i] < max) lst.erase(i);
190. }\*/
191. //lst = Delete\_List(lst, min, max);
192. int count = 0;
193. for (int i = 0; i < lst(); i++)
194. {
195. if (lst[i] > min && lst[i] < max) {
196. lst[i] = -1;
197. count++;
198. }
199. }
200. List<double> temp(lst() - count, 0);
201. int i = 0;
202. for (int j = 0; j < lst(); j++)
203. {
204. if (lst[j] != -1) {
205. temp[i] = lst[j];
206. i++;
207. }
208. }
209. lst = temp;
210. cout << "Полученный список: \n";
211. cout << lst << endl;
213. cout << "\nДобавление среднего арифметического к каждому элементу очереди:\n";
214. double middle = 0;
215. for (int i = 0; i < lst(); i++) middle += lst[i];
216. middle /= lst();
217. for (int i = 0; i < lst(); i++) lst[i] += middle;
218. cout << "Полученный список: \n";
219. cout << lst << endl;
220. }
222. void Task4()
223. {
224. int size;
225. cout << "Введите размер стека (1-100): "; cin >> size;
226. while (size < 1 || size > 100)
227. {
228. cout << "Ввод некорректен!\n";
229. cout << "Введите размер стека (1-100): "; cin >> size;
230. }
231. stack<double> st;
232. stack<double> tmp;
233. for (int i = 0; i < size; i++) st.push(rand() % 10000 / 100.0);
234. cout << "Сгенерированный стек: \n";
235. Print\_Stack(st);
237. cout << "\nДобавление максимума в конец стека\n";
238. tmp = st;
239. double max = 0;
240. for (int i = 0; i < size; i++)
241. {
242. if (tmp.top() > max) max = tmp.top();
243. tmp.pop();
244. }
245. for (int i = 0; i < size; i++)
246. {
247. tmp.push(st.top());
248. st.pop();
249. }
250. tmp.push(max);
251. size++;
252. for (int i = 0; i < size; i++)
253. {
254. st.push(tmp.top());
255. tmp.pop();
256. }
257. cout << "Полученный стек:\n";
258. Print\_Stack(st);
260. cout << "\nУдаление из списка чисел в заданном диапазоне\n";
261. double min;
262. cout << "Введите левую границу диапазона: "; cin >> min;
263. cout << "Введите правую границу диапазона: "; cin >> max;
264. cout << "Удаление из списка элементов в диапазоне [" << min << ";" << max << "]\n";
265. int count = 0;
266. for (int i = 0; i < size; i++)
267. {
268. if (st.top() < min || st.top() > max)
269. {
270. tmp.push(st.top());
271. count++;
272. }
273. st.pop();
274. }
275. size = count;
276. for (int i = 0; i < size; i++)
277. {
278. st.push(tmp.top());
279. tmp.pop();
280. }
281. cout << "Полученный стек:\n";
282. Print\_Stack(st);
284. cout << "\nДобавление среднего арифметического к каждому элементу стека\n";
285. double middle = 0;
286. for (int i = 0; i < size; i++)
287. {
288. middle += st.top();
289. tmp.push(st.top());
290. st.pop();
291. }
292. middle /= size;
293. for (int i = 0; i < size; i++)
294. {
295. st.push(tmp.top() + middle);
296. tmp.pop();
297. }
298. cout << "Полученный стек:\n";
299. Print\_Stack(st);
300. }
302. void Task5()
303. {
304. int size;
305. cout << "Введите размер стека (1-100): "; cin >> size;
306. while (size < 1 || size > 100)
307. {
308. cout << "Ввод некорректен!\n";
309. cout << "Введите размер стека (1-100): "; cin >> size;
310. }
311. stack<double> st;
312. stack<double> tmp;
313. for (int i = 0; i < size; i++) st.push(rand() % 10000 / 100.0);
314. cout << "Сгенерированный стек: \n";
315. Print\_Stack(st);
317. cout << "\nДобавление максимума в конец стека\n";
318. tmp = st;
319. double max = 0;
320. for (int i = 0; i < size; i++)
321. {
322. if (tmp.top() > max) max = tmp.top();
323. tmp.pop();
324. }
325. for (int i = 0; i < size; i++)
326. {
327. tmp.push(st.top());
328. st.pop();
329. }
330. tmp.push(max);
331. size++;
332. for (int i = 0; i < size; i++)
333. {
334. st.push(tmp.top());
335. tmp.pop();
336. }
337. cout << "Полученный стек:\n";
338. Print\_Stack(st);
340. cout << "\nУдаление из списка чисел в заданном диапазоне\n";
341. double min;
342. cout << "Введите левую границу диапазона: "; cin >> min;
343. cout << "Введите правую границу диапазона: "; cin >> max;
344. cout << "Удаление из списка элементов в диапазоне [" << min << ";" << max << "]\n";
345. int count = 0;
346. for (int i = 0; i < size; i++)
347. {
348. if (st.top() < min || st.top() > max)
349. {
350. tmp.push(st.top());
351. count++;
352. }
353. st.pop();
354. }
355. size = count;
356. for (int i = 0; i < size; i++)
357. {
358. st.push(tmp.top());
359. tmp.pop();
360. }
361. cout << "Полученный стек:\n";
362. Print\_Stack(st);
364. cout << "\nДобавление среднего арифметического к каждому элементу стека\n";
365. double middle = 0;
366. for (int i = 0; i < size; i++)
367. {
368. middle += st.top();
369. tmp.push(st.top());
370. st.pop();
371. }
372. middle /= size;
373. for (int i = 0; i < size; i++)
374. {
375. st.push(tmp.top() + middle);
376. tmp.pop();
377. }
378. cout << "Полученный стек:\n";
379. Print\_Stack(st);
380. }
382. int main()
383. {
384. setlocale(LC\_ALL, "Russian");
385. srand(time(0));
387. cout << "=========================| Двунаправленная очередь |=========================\n";
388. Task1();
389. cout << "\n=========================| Пара чисел |=========================\n";
390. Task2();
391. cout << "\n=========================| Список |=========================\n";
392. Task3();
393. cout << "\n=========================| Стек |=========================\n";
394. Task4();
396. return 0;
397. }

**Скриншоты результатов работы программы**

**Ответы на вопросы**

