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 №18.12**

Дисциплина: «Информатика»

Тема: “Объектно-ориентированное программирование.

Ассоциативные контейнеры библиотеки STL.”

Вариант 13

Выполнил:

Студент группы ИВТ-20-2б

Сафронов Владислав Владиславович

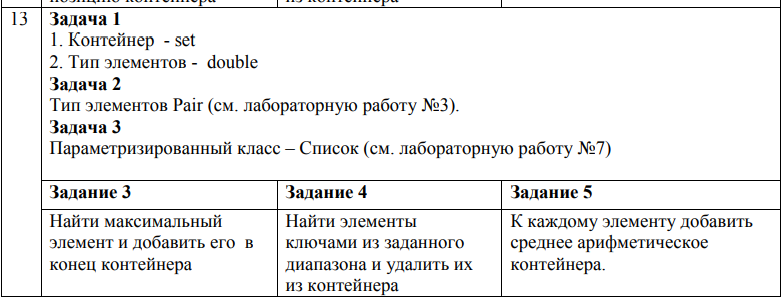
Проверила:

Доцент кафедры ИТАС

Полякова О.А.

**Пермь, 2021**

**Постановка задачи**

****

**Анализ задачи**

1. Для решения задачи необходимо:
   1. Использовать класс Pair.
   2. Использовать ассоциативные 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:
   1. typedef set<double> tset;

typedef set<Pair> pset;

* 1. tset m;

pset m;

* 1. List<double> task\_set(n);

1. Для операций ввода и вывода использовались следующие операторы и функции:
   1. Операторы << и >>.
2. Поставленные задачи будут решены следующими действиями:
   1. Функции для удаления элементов в заданном диапазоне:
3. void delete\_set(pset& m, Pair left, Pair right)
4. {
5. set<Pair>::iterator iter = m.begin();
6. vector<set<Pair>::iterator> iters;
7. int count = 0;
8. while (iter != m.end())
9. {
10. if (\*iter <= right && \*iter >= left) { count++; iters.push\_back(iter); }
11. iter++;
12. }
13. cout << "Элементов к удалению: " << count << endl;
14. for (int i = 0; i < count; i++)
15. {
16. m.erase(iters[i]);
17. }
18. }
19. void delete\_set(tset& m, double left, double right)
20. {
21. set<double>::iterator iter = m.begin();
22. vector<set<double>::iterator> iters;
23. int count = 0;
24. while (iter != m.end())
25. {
26. if (\*iter <= right && \*iter >= left) { count++; iters.push\_back(iter); }
27. iter++;
28. }
29. cout << "Элементов к удалению: " << count << endl;
30. for (int i = 0; i < count; i++)
31. {
32. m.erase(iters[i]);
33. }
34. }
    1. Функции для добавления среднего арифметического к каждому элементу контейнера:
35. void add\_average(tset& m)
36. {
37. set<double>::iterator iter = m.begin();
38. tset temp\_set;
39. int size = 0;
40. double avg = 0;
41. while (iter != m.end())
42. {
43. avg += \*iter;
44. size++;
45. iter++;
46. }
47. avg /= size;
48. iter = m.begin();
49. while (iter != m.end())
50. {
51. double tmp = (\*iter) + avg;
52. iter++;
53. temp\_set.insert(tmp);
54. }
55. m = temp\_set;
56. }
57. void add\_average(pset& m)
58. {
59. set<Pair>::iterator iter = m.begin();
60. pset temp\_set;
61. int size = 0;
62. Pair avg;
63. while (iter != m.end())
64. {
65. avg += \*iter;
66. size++;
67. iter++;
68. }
69. avg /= size;
70. iter = m.begin();
71. while (iter != m.end())
72. {
73. Pair tmp = avg + (\*iter);
74. iter++;
75. temp\_set.insert(tmp);
76. }
77. m = temp\_set;
78. }
79. void add\_average(List<double>& m)
80. {
81. double avg = 0;
82. for (int i = 0; i < m(); i++)
83. {
84. avg += m[i];
85. }
86. avg /= m();
87. List<double> tmp = m + avg;
88. m = tmp;
89. }

**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. }

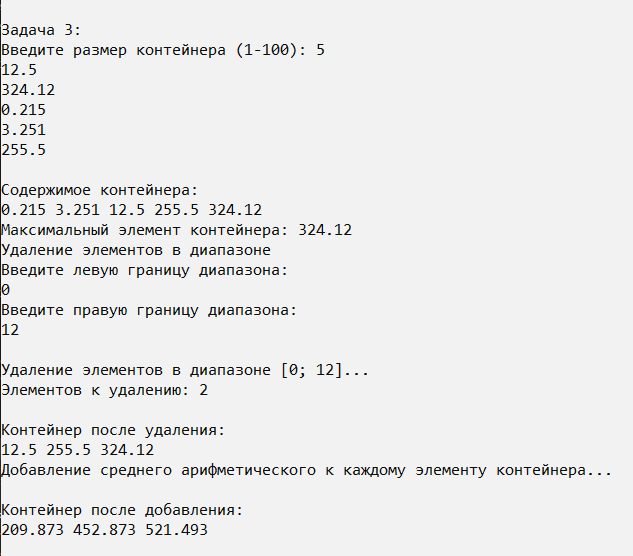
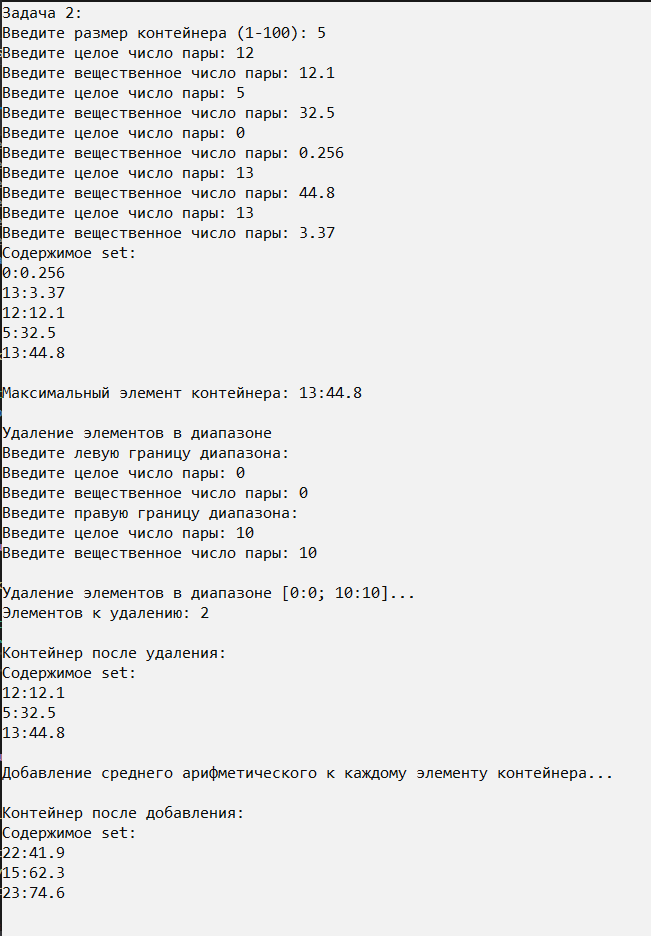
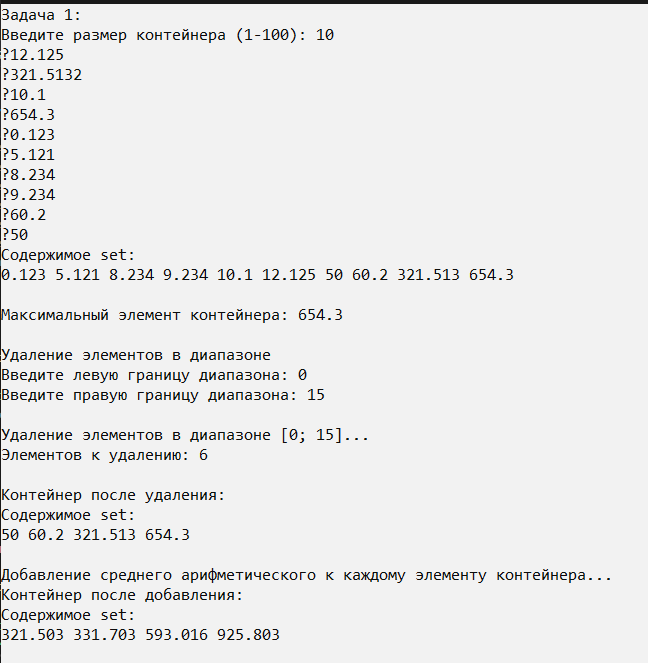
**Файл Lish.h:**

1. #pragma once
2. #include <iostream>
3. #include <set>
4. using namespace std;
6. template <class T>
7. class List
8. {
9. int size;
10. set<T> data;
12. public:
13. List(int s);
14. List(const List<T>& a);
15. ~List();
17. List& operator = (const List<T>& a);
18. T& operator [] (int index);
19. List operator + (const T k);
20. List operator + (const List<T>& a);
21. void push\_back(T a);
22. void erase(int pos);
23. int operator () ();
24. void Fill();
25. void Show();
26. void Delete\_task(T left, T right);
27. void Add\_Average();
28. //friend ostream& operator << <>(ostream& out, const List<T>& a);
29. //friend istream& operator >> <>(istream& in, List<T>& a);
30. };
32. template <class T>
33. List<T>::List(int s)
34. {
35. size = s;
36. }
37. template <class T>
38. List<T>::List(const List& a)
39. {
40. size = a.size;
41. data = a.data;
42. }
43. template <class T>
44. List<T>::~List()
45. {
46. }
47. template <class T>
48. void List<T>::push\_back(T a)
49. {
50. data.insert(a);
51. size++;
52. }
53. template <class T>
54. void List<T>::erase(int pos)
55. {
56. set<double>::iterator iter = data.begin();
57. //iterator iter = data.begin();
58. for (int i; i < pos; i++) iter++;
59. data.erase(iter);
60. size--;
61. }
62. template<class T>
63. List<T>& List<T>::operator=(const List<T>& a)
64. {
65. if (this == &a) return \*this;
66. size = a.size;
67. data = a.data;
68. return \*this;
69. }
70. template<class T>
71. T& List<T>::operator[](int index)
72. {
74. if (index < size) {
75. set<double>::iterator iter = data.begin();
76. //iterator iter = data.begin();
77. for (int i = 0; i < index; i++) iter++;
78. double tmp = \*iter;
79. return tmp;
80. }
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);
87. set<T> temp\_set;
88. for (int i = 0; i < size; ++i) temp\_set.insert((\*this)[i] + k);
89. temp.data = temp\_set;
90. return temp;
91. }
92. template<class T>
93. List<T> List<T>::operator+(const List<T>& a)
94. {
95. int min = size < a.size ? size : a.size;
96. List<T> temp(min, 0);
97. set<T> temp\_set;
98. for (int i = 0; i < min; i++) temp\_set.insert(data[i] + a.data[i]);
99. temp.data = temp\_set;
100. return temp;
101. }
102. template<class T>
103. int List<T>::operator () ()
104. {
105. return size;
106. }
107. template<class T>
108. void List<T>::Show()
109. {
110. set<double>::iterator iter = data.begin();
111. while (iter != data.end())
112. {
113. cout << (\*iter) << " ";
114. iter++;
115. }
116. //for (; iter != data.end(); iter++) cout << (\*iter) << " ";
117. }
118. template<class T>
119. void List<T>::Fill()
120. {
121. data.clear();
122. T tmp;
123. for (int i = 0; i < size; ++i) { cin >> tmp; data.insert(tmp); }
124. }
125. template<class T>
126. void List<T>::Delete\_task(T left, T right)
127. {
128. set<double>::iterator iter = data.begin();
129. vector<set<double>::iterator> iters;
130. int count = 0;
131. while (iter != data.end())
132. {
133. if (\*iter <= right && \*iter >= left) { count++; iters.push\_back(iter); }
134. iter++;
135. }
136. cout << "Ёлементов к удалению: " << count << endl;
137. for (int i = 0; i < count; i++)
138. {
139. data.erase(iters[i]);
140. size--;
141. }
142. }
143. template<class T>
144. void List<T>::Add\_Average()
145. {
146. double avg = 0;
147. for (int i = 0; i < size; i++)
148. {
149. avg += (\*this)[i];
150. }
151. avg /= size;
152. set<T> tmp;
153. for (int i = 0; i < size; i++)
154. {
155. tmp.insert(((\*this)[i] + avg));
156. }
157. data = tmp;
158. }

**Файл main.cpp:**

1. #include <iostream>
2. #include <map>
3. #include <vector>
4. #include <set>
5. #include "Pair.h"
6. #include "List.h"
7. using namespace std;
9. typedef set<double> tset;
10. typedef set<Pair> pset;
12. tset make\_set(int n)
13. {
14. tset m;
15. double a;
16. for (int i = 0; i < n; i++)
17. {
18. cout << "?";
19. cin >> a;
20. m.insert(a);
21. }
22. return m;
23. }
24. pset make\_pair\_set(int n)
25. {
26. pset m;
27. Pair a;
28. for (int i = 0; i < n; i++)
29. {
30. cin >> a;
31. m.insert(a);
32. }
33. return m;
34. }
36. void print\_set(tset m)
37. {
38. cout << "Содержимое set: \n";
39. copy(m.begin(), m.end(), ostream\_iterator<double>(cout, " "));
40. cout << endl;
41. }
42. void print\_set(pset m)
43. {
44. cout << "Содержимое set: \n";
45. set<Pair>::iterator iter = m.begin();
46. for (; iter != m.end(); iter++)
47. {
48. cout << \*iter << endl;
49. }
50. //copy(m.begin(), m.end(), ostream\_iterator<Pair>(cout, " "));
51. cout << endl;
52. }
54. void delete\_set(tset& m, double left, double right)
55. {
56. set<double>::iterator iter = m.begin();
57. vector<set<double>::iterator> iters;
58. int count = 0;
59. while (iter != m.end())
60. {
61. if (\*iter <= right && \*iter >= left) { count++; iters.push\_back(iter); }
62. iter++;
63. }
64. cout << "Элементов к удалению: " << count << endl;
65. for (int i = 0; i < count; i++)
66. {
67. m.erase(iters[i]);
68. }
69. }
70. void delete\_set(pset& m, Pair left, Pair right)
71. {
72. set<Pair>::iterator iter = m.begin();
73. vector<set<Pair>::iterator> iters;
74. int count = 0;
75. while (iter != m.end())
76. {
77. if (\*iter <= right && \*iter >= left) { count++; iters.push\_back(iter); }
78. iter++;
79. }
80. cout << "Элементов к удалению: " << count << endl;
81. for (int i = 0; i < count; i++)
82. {
83. m.erase(iters[i]);
84. }
85. }
87. void add\_average(tset& m)
88. {
89. set<double>::iterator iter = m.begin();
90. tset temp\_set;
91. int size = 0;
92. double avg = 0;
93. while (iter != m.end())
94. {
95. avg += \*iter;
96. size++;
97. iter++;
98. }
99. avg /= size;
100. iter = m.begin();
101. while (iter != m.end())
102. {
103. double tmp = (\*iter) + avg;
104. iter++;
105. temp\_set.insert(tmp);
106. }
107. m = temp\_set;
108. }
109. void add\_average(pset& m)
110. {
111. set<Pair>::iterator iter = m.begin();
112. pset temp\_set;
113. int size = 0;
114. Pair avg;
115. while (iter != m.end())
116. {
117. avg += \*iter;
118. size++;
119. iter++;
120. }
121. avg /= size;
122. iter = m.begin();
123. while (iter != m.end())
124. {
125. Pair tmp = avg + (\*iter);
126. iter++;
127. temp\_set.insert(tmp);
128. }
129. m = temp\_set;
130. }
131. void add\_average(List<double>& m)
132. {
133. double avg = 0;
134. for (int i = 0; i < m(); i++)
135. {
136. avg += m[i];
137. }
138. avg /= m();
139. List<double> tmp = m + avg;
140. m = tmp;
141. }
143. void task1()
144. {
145. int n; double left, right;
146. cout << "Введите размер контейнера (1-100): "; cin >> n;
147. while (n < 1 || n > 100)
148. {
149. cout << "\nВвод некорректен!";
150. cout << "\nВведите размер контейнера (1-100): "; cin >> n;
151. }
152. tset m = make\_set(n);
153. print\_set(m);
155. cout << "\nМаксимальный элемент контейнера: ";
156. set<double>::iterator iter = m.end(); iter--;
157. double tmp = \*iter;
158. cout << tmp << endl;
160. cout << "\nУдаление элементов в диапазоне";
161. cout << "\nВведите левую границу диапазона: "; cin >> left;
162. cout << "Введите правую границу диапазона: "; cin >> right;
163. cout << "\nУдаление элементов в диапазоне [" << left << "; " << right << "]...\n";
164. delete\_set(m, left, right);
165. cout << "\nКонтейнер после удаления:\n";
166. print\_set(m);
167. cout << "\nДобавление среднего арифметического к каждому элементу контейнера...";
168. add\_average(m);
169. cout << "\nКонтейнер после добавления:\n";
170. print\_set(m);
171. }
172. void task2()
173. {
174. int n; Pair left, right;
175. cout << "Введите размер контейнера (1-100): "; cin >> n;
176. while (n < 1 || n > 100)
177. {
178. cout << "\nВвод некорректен!";
179. cout << "\nВведите размер контейнера (1-100): "; cin >> n;
180. }
181. pset m = make\_pair\_set(n);
182. print\_set(m);
184. cout << "Максимальный элемент контейнера: ";
185. set<Pair>::iterator iter = m.end(); iter--;
186. Pair tmp = \*iter;
187. cout << tmp << endl;
188. cout << "\nУдаление элементов в диапазоне";
189. cout << "\nВведите левую границу диапазона: \n"; cin >> left;
190. cout << "Введите правую границу диапазона: \n"; cin >> right;
191. cout << "\nУдаление элементов в диапазоне [" << left << "; " << right << "]...\n";
192. delete\_set(m, left, right);
193. cout << "\nКонтейнер после удаления:\n";
194. print\_set(m);
195. cout << "Добавление среднего арифметического к каждому элементу контейнера...\n";
196. add\_average(m);
197. cout << "\nКонтейнер после добавления:\n";
198. print\_set(m);
199. }
200. void task3()
201. {
202. int n; double left, right;
203. cout << "Введите размер контейнера (1-100): "; cin >> n;
204. while (n < 1 || n > 100)
205. {
206. cout << "\nВвод некорректен!";
207. cout << "\nВведите размер контейнера (1-100): "; cin >> n;
208. }
209. List<double> task\_set(n);
210. task\_set.Fill();
211. cout << "\nСодержимое контейнера: \n";
212. task\_set.Show();
213. cout << endl;
214. cout << "Максимальный элемент контейнера: " << task\_set[n-1];
215. cout << "\nУдаление элементов в диапазоне";
216. cout << "\nВведите левую границу диапазона: \n"; cin >> left;
217. cout << "Введите правую границу диапазона: \n"; cin >> right;
218. cout << "\nУдаление элементов в диапазоне [" << left << "; " << right << "]...\n";
219. task\_set.Delete\_task(left, right);
220. cout << "\nКонтейнер после удаления:\n";
221. task\_set.Show();
222. cout << endl;
223. cout << "Добавление среднего арифметического к каждому элементу контейнера...\n";
224. //add\_average(task\_set);
225. task\_set.Add\_Average();
226. cout << "\nКонтейнер после добавления:\n";
227. task\_set.Show();
228. cout << endl;
229. }
231. int main()
232. {
233. setlocale(LC\_ALL, "Russian");
234. cout << "Задача 1:\n";
235. task1();
236. cout << "\nЗадача 2:\n";
237. task2();
238. cout << "\nЗадача 3:\n";
239. task3();
241. return 0;
242. }

**Скриншоты результатов работы программы**

****

**Ответы на вопросы**

