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 №18.7**

Дисциплина: «Информатика»

Тема: “Объектно-ориентированное программирование.

Шаблоны классов.”

Вариант 13

Выполнил:

Студент группы ИВТ-20-2б

Сафронов Владислав Владиславович

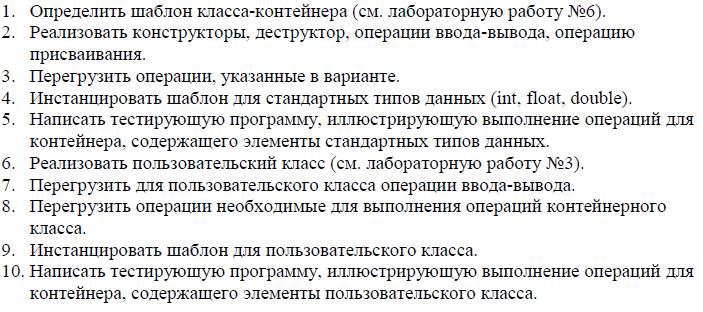
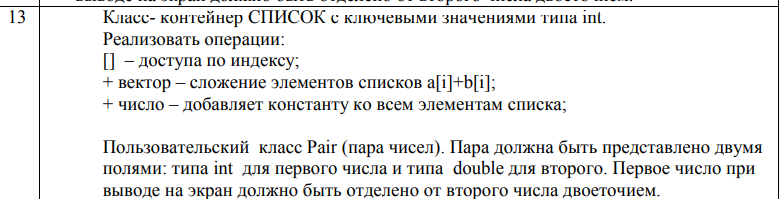
Проверила:

Доцент кафедры ИТАС

Полякова О.А.

**Пермь, 2021**

**Постановка задачи**

****

**Анализ задачи**

1. Для решения задачи необходимо:
   1. Реализовать шаблонный класс List.
   2. Реализовать перегрузки операторов из варианта.
2. В ходе работы были использованы следующие типы данных:
   1. Класс Pair из лабораторной работы №2.
   2. Шаблонный класс List
3. template <class T>
4. class List
5. {
6. T\* data = nullptr;
7. int size;
   * 1. Конструкторы класса List
8. List()
9. {
10. data = nullptr;
11. size = 0;
12. }
13. List(int s)
14. {
15. size = s;
16. data = new T[size];
17. }
18. List(List<T>& tmp)
19. {
20. if (tmp.get\_size() == 0) List();
21. else
22. {
23. size = tmp.get\_size();
24. data = new T[size];
25. for (int i = 0; i < size; i++)
26. {
27. data[i] = tmp[i];
28. }
29. }
30. }
    * 1. Деструктор класса List
31. ~List()
32. {
33. delete[] data;
34. data = nullptr;
35. size = 0;
36. }
    * 1. Метод get\_size возвращает размер списка.
37. int get\_size() { return size; }
    * 1. Метод empty проверяет список на пустоту. Если список пуст, возвращается true, иначе – false.
38. bool empty() { return size == 0; }
    * 1. Метод push\_back для добавления элемента в список.
39. void push\_back(T elem)
40. {
41. T\* tmp = new T[size + 1];
42. for (int i = 0; i < size; i++)
43. {
44. tmp[i] = data[i];
45. }
46. tmp[size] = elem;
47. size++;
48. delete[] data;
49. data = tmp;
50. tmp = nullptr;
51. }
    * 1. Метод pop\_back для удаления последнего элемента списка.
52. void pop\_back()
53. {
54. if (empty()) return;
55. T\* tmp = new T[size - 1];
56. for (int i = 0; i < size; i++)
57. {
58. tmp[i] = data[i];
59. }
60. delete[] data;
61. data = tmp;
62. size--;
63. tmp = nullptr;
64. }
    * 1. Метод erase для удаления элемента из списка по индексу.
65. void erase(int index)
66. {
67. if (empty()) return;
68. T\* tmp = new T[size - 1];
69. for (int i = 0; i < index; i++)
70. {
71. tmp[i] = data[i];
72. }
73. for (int i = index; i < size - 1; i++)
74. {
75. tmp[i] = data[i + 1];
76. }
77. delete[] data;
78. data = tmp;
79. size--;
80. tmp = nullptr;
81. }
    * 1. Перегрузка оператора []
82. T& operator[](int index) { return data[index]; }
    * 1. Перегрузки математических операций
83. List<T>& operator=(List<T>& tmp)
84. {
85. delete[] data;
86. size = tmp.get\_size();
87. data = new T[size];
88. for (int i = 0; i < size; i++)
89. {
90. data[i] = tmp[i];
91. }
92. return \*this;
93. }
95. List<T>& operator+(List<T>& tmp)
96. {
97. if (empty() || tmp.empty()) return \*this;
98. if (size > tmp.get\_size())
99. {
100. for (int i = 0; i < tmp.get\_size(); i++)
101. {
102. data[i] = data[i] + tmp[i];
103. }
104. }
105. else
106. {
107. for (int i = 0; i < size; i++)
108. {
109. data[i] = data[i] + tmp[i];
110. }
111. }
112. return \*this;
113. }
114. List<T>& operator+(T tmp)
115. {
116. if (empty()) return \*this;
117. for (int i = 0; i < size; i++)
118. {
119. data[i] = data[i] + tmp;
120. }
121. return \*this;
122. }
     * 1. Перегрузка операторов << и >>
123. template <class T>
124. istream& operator>>(istream& in, List<T>& tmp)
125. {
126. for (int i = 0; i < tmp.get\_size(); i++)
127. {
128. cout << "Введите " << i + 1 << " элемент спика: ";
129. in >> tmp.data[i];
130. }
131. return in;
132. }
134. template <class T>
135. ostream& operator<<(ostream& out, List<T>& tmp)
136. {
137. for (int i = 0; i < tmp.get\_size(); i++) out << tmp.data[i] << "\n";
138. return out;
139. }
140. Описание функции main()
     1. Создается список типа int размерностью 10 и с клавиатуры вводятся его элементы. Затем список выводится в консоль.
141. int main()
142. {
143. setlocale(LC\_ALL, "rus");
144. srand(time(0));
146. List<int> lst\_1(10);
147. cout << "Ввод списка 1: \n";
148. cin >> lst\_1;
149. cout << "Вывод списка 1: \n";
150. cout << lst\_1;
     1. Создается второй список. Ему присваивается значения списка 1 и ко всем элементам прибавляется 5.
151. List<int> lst\_2;
152. lst\_2 = lst\_1 + 5;
153. cout << "Вывод списка 2(список 1 + 5): \n";
154. cout << lst\_2;
     1. Создактся третий список, который является результатом суммы первого и второго списков.
155. List<int> lst\_3;
156. lst\_3 = lst\_1 + lst\_2;
157. cout << "Вывод списка 3(список 1 + список 2): \n";
158. cout << lst\_3;
     1. Демонстрация работы класса с элементами пользовательского класса Pair. Создается список типа Pair размерностью 3. Вводятся значения элементов списка, затем список выводится в консоль.
159. List<Pair> lst\_4(3);
160. cout << "Ввод списка Pair: \n";
161. cin >> lst\_4;
162. cout << "Вывод списка Pair: \n";
163. cout << lst\_4;

**Блок-схема программы**

**Код программы**

**Файл List.h:**

1. #pragma once
2. #include "Pair.h"
3. #include <iostream>
4. using namespace std;
6. template <class T>
7. class List
8. {
9. T\* data = nullptr;
10. int size;
11. public:
12. List()
13. {
14. data = nullptr;
15. size = 0;
16. }
17. List(int s)
18. {
19. size = s;
20. data = new T[size];
21. }
22. List(List<T>& tmp)
23. {
24. if (tmp.get\_size() == 0) List();
25. else
26. {
27. size = tmp.get\_size();
28. data = new T[size];
29. for (int i = 0; i < size; i++)
30. {
31. data[i] = tmp[i];
32. }
33. }
34. }
35. ~List()
36. {
37. delete[] data;
38. data = nullptr;
39. size = 0;
40. }
42. int get\_size() { return size; }
44. bool empty() { return size == 0; }
46. void push\_back(T elem)
47. {
48. T\* tmp = new T[size + 1];
49. for (int i = 0; i < size; i++)
50. {
51. tmp[i] = data[i];
52. }
53. tmp[size] = elem;
54. size++;
55. delete[] data;
56. data = tmp;
57. tmp = nullptr;
58. }
59. void pop\_back()
60. {
61. if (empty()) return;
62. T\* tmp = new T[size - 1];
63. for (int i = 0; i < size; i++)
64. {
65. tmp[i] = data[i];
66. }
67. delete[] data;
68. data = tmp;
69. size--;
70. tmp = nullptr;
71. }
72. void erase(int index)
73. {
74. if (empty()) return;
75. T\* tmp = new T[size - 1];
76. for (int i = 0; i < index; i++)
77. {
78. tmp[i] = data[i];
79. }
80. for (int i = index; i < size - 1; i++)
81. {
82. tmp[i] = data[i + 1];
83. }
84. delete[] data;
85. data = tmp;
86. size--;
87. tmp = nullptr;
88. }
90. T& operator[](int index) { return data[index]; }
92. List<T>& operator=(List<T>& tmp)
93. {
94. delete[] data;
95. size = tmp.get\_size();
96. data = new T[size];
97. for (int i = 0; i < size; i++)
98. {
99. data[i] = tmp[i];
100. }
101. return \*this;
102. }
104. List<T>& operator+(List<T>& tmp)
105. {
106. if (empty() || tmp.empty()) return \*this;
107. if (size > tmp.get\_size())
108. {
109. for (int i = 0; i < tmp.get\_size(); i++)
110. {
111. data[i] = data[i] + tmp[i];
112. }
113. }
114. else
115. {
116. for (int i = 0; i < size; i++)
117. {
118. data[i] = data[i] + tmp[i];
119. }
120. }
121. return \*this;
122. }
123. List<T>& operator+(T tmp)
124. {
125. if (empty()) return \*this;
126. for (int i = 0; i < size; i++)
127. {
128. data[i] = data[i] + tmp;
129. }
130. return \*this;
131. }
133. friend istream& operator>> <>(istream& in, List<T>& tmp);
134. friend ostream& operator<< <>(ostream& out, List<T>& tmp);
135. };
137. template <class T>
138. istream& operator>>(istream& in, List<T>& tmp)
139. {
140. for (int i = 0; i < tmp.get\_size(); i++)
141. {
142. cout << "Введите " << i + 1 << " элемент спика: ";
143. in >> tmp.data[i];
144. }
145. return in;
146. }
148. template <class T>
149. ostream& operator<<(ostream& out, List<T>& tmp)
150. {
151. for (int i = 0; i < tmp.get\_size(); i++) out << tmp.data[i] << "\n";
152. return out;
153. }

**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 main.cpp:**

1. #include <iostream>
2. #include <ctime>
3. #include "List.h"
4. #include "Pair.h"
5. using namespace std;
7. int main()
8. {
9. setlocale(LC\_ALL, "rus");
10. srand(time(0));
12. List<int> lst\_1(10);
13. cout << "Ввод списка 1: \n";
14. cin >> lst\_1;
15. cout << "Вывод списка 1: \n";
16. cout << lst\_1;
17. List<int> lst\_2;
18. lst\_2 = lst\_1 + 5;
19. cout << "Вывод списка 2(список 1 + 5): \n";
20. cout << lst\_2;
21. List<int> lst\_3;
22. lst\_3 = lst\_1 + lst\_2;
23. cout << "Вывод списка 3(список 1 + список 2): \n";
24. cout << lst\_3;
26. List<Pair> lst\_4(3);
27. cout << "Ввод списка Pair: \n";
28. cin >> lst\_4;
29. cout << "Вывод списка Pair: \n";
30. cout << lst\_4;
31. }

**Скриншоты результатов работы программы**

**Ответы на вопросы**

