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 №18.3**

Дисциплина: «Информатика»

Тема: Перегрузка операций.

Вариант 13

Выполнил:

Студент группы ИВТ-20-2б

Сафронов Владислав Владиславович

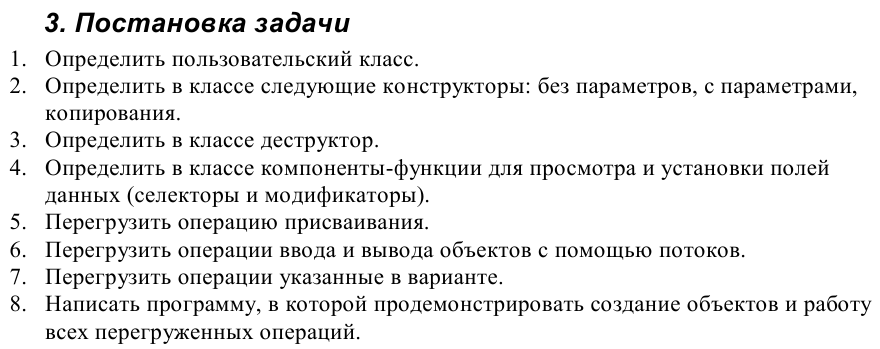
Проверила:

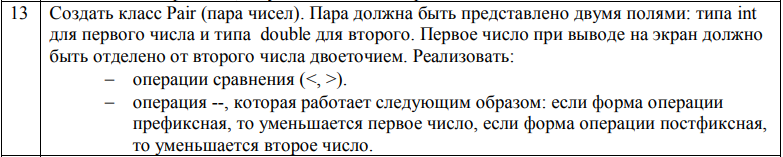
Доцент кафедры ИТАС

Полякова О.А.

**Пермь, 2021**

**Постановка задачи**



****

**Анализ задачи**

1. Для решения задачи необходимо:
   1. Разработать класс Pair для хранения пары целого (first) и вещественного (second) числа.
   2. Организовать необходимые методы для ввода данных в поля first и second, конструкторы, деструктор.
   3. Организовать перегрузку операторов сравнения
   4. Организовать перегрузку операторов постфиксного и префиксного декремента.
2. В ходе работы были использованы следующие типы данных:
   1. Класс Pair для хранения челого числа (first) и вещественного числа (second).
3. class Pair
4. {
5. public:
6. int first;
7. double second;
   1. Конструкторы класса Pair: без параметров, с параметрами, а также конструктор копирования.
8. Pair()
9. {
10. first = 0;
11. second = 0;
12. }
14. Pair(int f, double s)
15. {
16. this->first = f;
17. this->second = s;
18. }
19. Pair(const Pair& obj)
20. {
21. first = obj.first;
22. second = obj.second;
23. }
    1. Метод Show для вывода содержимого полей в консоль.
24. void Show()
25. {
26. cout << first << ":" << second << endl;
27. }
    1. Методы SetFirst и SetSecond устанавливают переданные значения в соответствующие поля.
28. void SetFirst(int n)
29. {
30. first = n;
31. }
32. void SetSecond(double n)
33. {
34. second = n;
35. }
    1. Перегрузка оперций постфиксного и префиксного декремента. Если форма префиксная – уменьшается первое число, если постфиксная – второе.
36. Pair& operator--()
37. {
38. first--;
39. return \*this;
40. }
41. Pair operator--(int)
42. {
43. second--;
44. return \*this;
45. }
    1. Перегрузка операторов сравнения для класса Pair.
46. bool operator>(const Pair& obj1, const Pair& obj2)
47. {
48. //return (obj1.first > obj2.first && obj1.second > obj2.second);
49. return (((double)obj1.first + obj1.second) > ((double)obj2.first + obj2.second));
50. }
52. bool operator>=(const Pair& obj1, const Pair& obj2)
53. {
54. //return (obj1.first >= obj2.first && obj1.second >= obj2.second);
55. return (((double)obj1.first + obj1.second) >= ((double)obj2.first + obj2.second));
56. }
58. bool operator<(const Pair& obj1, const Pair& obj2)
59. {
60. //return (obj1.first < obj2.first && obj1.second < obj2.second);
61. return (((double)obj1.first + obj1.second) < ((double)obj2.first + obj2.second));
62. }
64. bool operator<=(const Pair& obj1, const Pair& obj2)
65. {
66. //return (obj1.first <= obj2.first && obj1.second <= obj2.second);
67. return (((double)obj1.first + obj1.second) <= ((double)obj2.first + obj2.second));
68. }
70. bool operator==(const Pair& obj1, const Pair& obj2)
71. {
72. return (obj1.first == obj2.first && obj1.second == obj2.second);
73. }
75. bool operator!=(const Pair& obj1, const Pair& obj2)
76. {
77. return (obj1.first != obj2.first || obj1.second != obj2.second);
78. }
79. Описание функции main()
    1. Создается 2 экземпляра класса Pair с параметрами, затем оба выводятся в консоль с помощью метода Show.
80. Pair a(8, 3.14);
81. Pair b(9, 2.45);
83. cout << "a: ";
84. a.Show();
86. cout << "b: ";
87. a.Show();
89. cout << "\n--a: ";
90. (--a).Show();
92. cout << "a--: ";
93. (a--).Show();
    1. Демонстрация работы операторов сравнения для класса Pair
94. cout << endl << "a > b: " << (a > b) << endl;
95. cout << "a >= b: " << (a >= b) << endl;
96. cout << "a < b: " << (a < b) << endl;
97. cout << "a <= b: " << (a <= b) << endl;

**Блок-схема программы**

**Код программы**

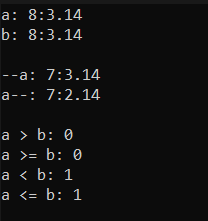
**Файл Pair.h:**

1. #pragma once
2. #include <iostream>
3. using namespace std;
5. class Pair
6. {
7. public:
8. int first;
9. double second;
10. public:
11. Pair()
12. {
13. first = 0;
14. second = 0;
15. }
17. Pair(int f, double s)
18. {
19. this->first = f;
20. this->second = s;
21. }
22. Pair(const Pair& obj)
23. {
24. first = obj.first;
25. second = obj.second;
26. }
28. void Show()
29. {
30. cout << first << ":" << second << endl;
31. }
33. void SetFirst(int n)
34. {
35. first = n;
36. }
37. void SetSecond(double n)
38. {
39. second = n;
40. }
42. friend bool operator>(const Pair& obj1, const Pair& obj2);
43. friend bool operator>=(const Pair& obj1, const Pair& obj2);
44. friend bool operator<(const Pair& obj1, const Pair& obj2);
45. friend bool operator<=(const Pair& obj1, const Pair& obj2);
46. Pair& operator--() // �������
47. {
48. first--;
49. return \*this;
50. }
51. Pair operator--(int) //��������
52. {
53. second--;
54. return \*this;
55. }
57. ~Pair() {};
58. };
60. bool operator>(const Pair& obj1, const Pair& obj2)
61. {
62. //return (obj1.first > obj2.first && obj1.second > obj2.second);
63. return (((double)obj1.first + obj1.second) > ((double)obj2.first + obj2.second));
64. }
66. bool operator>=(const Pair& obj1, const Pair& obj2)
67. {
68. //return (obj1.first >= obj2.first && obj1.second >= obj2.second);
69. return (((double)obj1.first + obj1.second) >= ((double)obj2.first + obj2.second));
70. }
72. bool operator<(const Pair& obj1, const Pair& obj2)
73. {
74. //return (obj1.first < obj2.first && obj1.second < obj2.second);
75. return (((double)obj1.first + obj1.second) < ((double)obj2.first + obj2.second));
76. }
78. bool operator<=(const Pair& obj1, const Pair& obj2)
79. {
80. //return (obj1.first <= obj2.first && obj1.second <= obj2.second);
81. return (((double)obj1.first + obj1.second) <= ((double)obj2.first + obj2.second));
82. }
84. bool operator==(const Pair& obj1, const Pair& obj2)
85. {
86. return (obj1.first == obj2.first && obj1.second == obj2.second);
87. }
89. bool operator!=(const Pair& obj1, const Pair& obj2)
90. {
91. return (obj1.first != obj2.first || obj1.second != obj2.second);
92. }

**Файл main.cpp:**

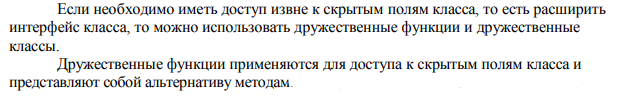
1. #include <iostream>
2. #include "Pair.h"
3. using namespace std;
5. int main()
6. {
7. Pair a(8, 3.14);
8. Pair b(9, 2.45);
10. cout << "a: ";
11. a.Show();
13. cout << "b: ";
14. a.Show();
16. cout << "\n--a: ";
17. (--a).Show();
19. cout << "a--: ";
20. (a--).Show();
22. cout << endl << "a > b: " << (a > b) << endl;
23. cout << "a >= b: " << (a >= b) << endl;
24. cout << "a < b: " << (a < b) << endl;
25. cout << "a <= b: " << (a <= b) << endl;
26. }

**Скриншоты результатов работы программы**

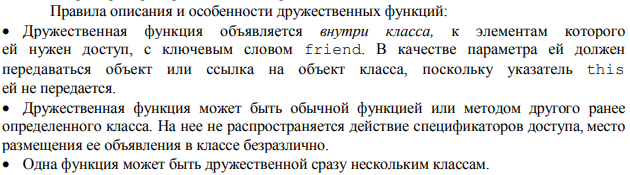
****

**Ответы на вопросы**



1. 

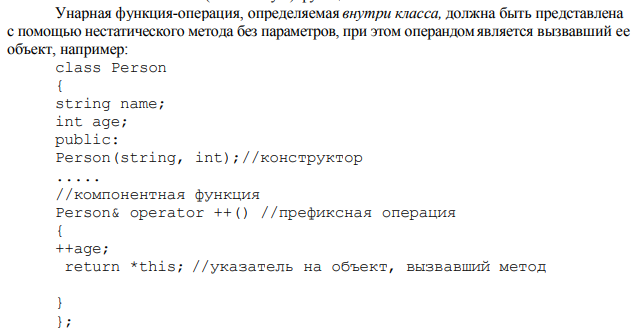


2. 

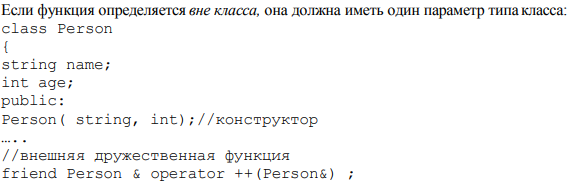


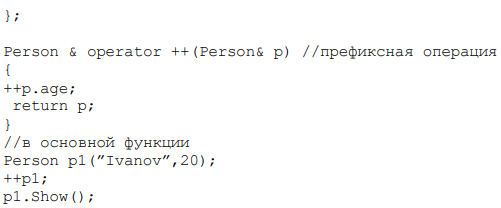
3. 



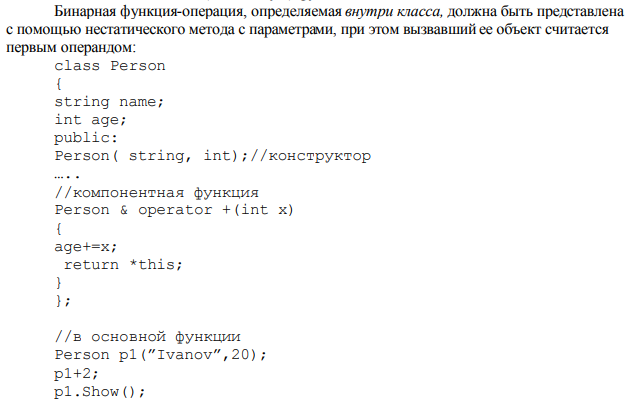
4.



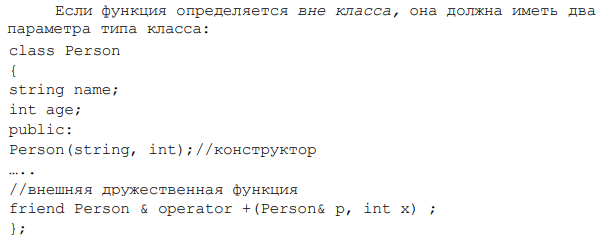
5.





6.

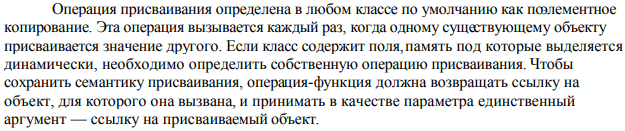


7. 



8. 

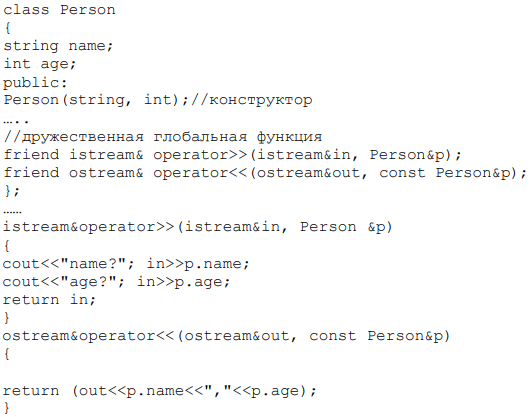


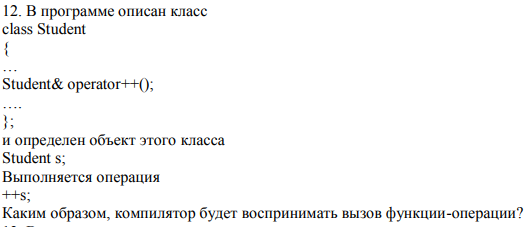
9. 



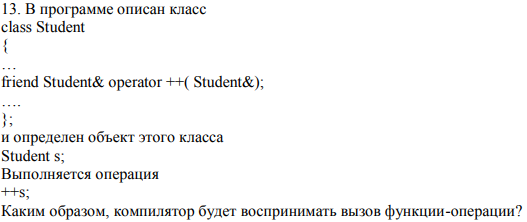
10. Ссылку на объект, для которого операция была вызвана.



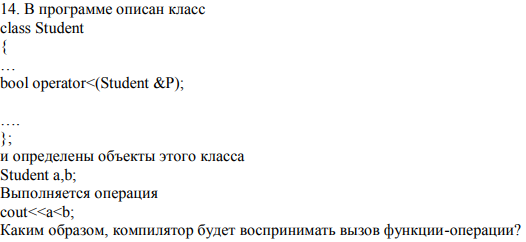




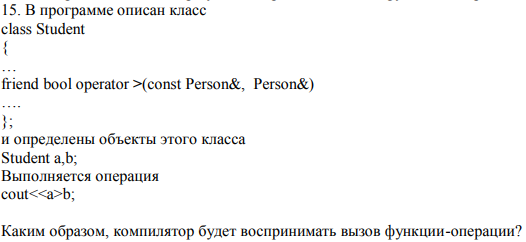
12. Будет осуществлён вызов функции перегрузки оператора префиксного инкремента.



13. Будет осуществлён вызов функции перегрузки оператора постфиксного инкремента.



14. Будет осуществлён вызов функции перегрузки оператора сравнения (<), затем через cout будет выведен результат сравнения (1 или 0).



15. Будет осуществлён вызов функции перегрузки оператора сравнения (>), затем через cout будет выведен результат сравнения (1 или 0).