ГУАП

КАФЕДРА № 43

ОТЧЕТ

ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

старший преподаватель E.О. Шумова

должность, уч. степень, звание подпись, дата инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ

«МНОЖЕСТВЕННОЕ НАСЛЕДОВАНИЕ В ЯЗЫКЕ C++»

по курсу: ОБЪЕКТНО ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ

РАБОТУ ВЫПОЛНИЛ

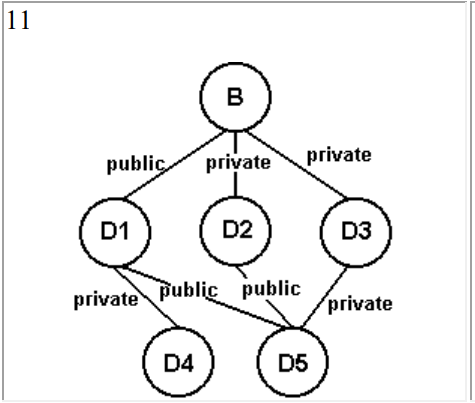
СТУДЕНТ ГР. № 4131 Иванов И.В.

подпись, дата инициалы, фамилия

Санкт-Петербург 2022

**Цель работы:** Получение практических навыков при использовании множественного наследования в языке С++.

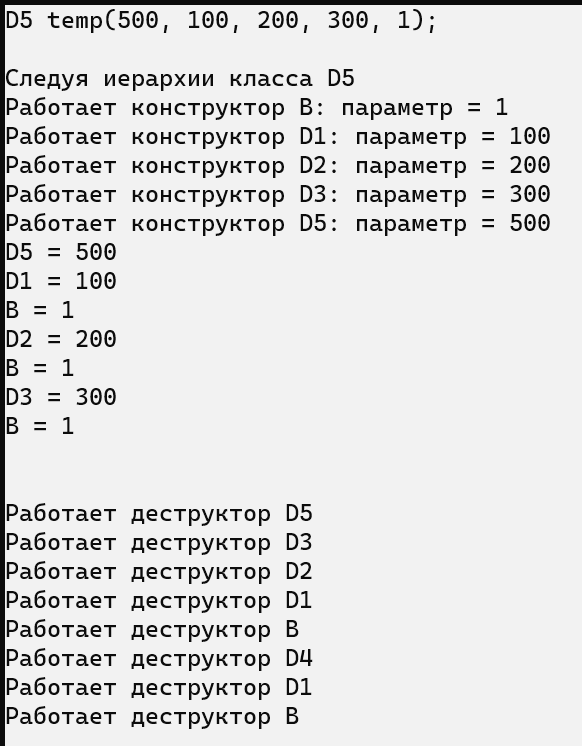
**Условие для варианта:** 11



**Листинг программы:**

1. #include <iostream>
3. using namespace std;
5. class B {
6. int value;
8. public:
9. B() {
10. cout << "Работает конструктор по умолчанию класса B" << "\n";
11. };
13. B(int v):value(v) {
14. cout << "Работает конструктор B: параметр = " << value << "\n";
15. };
17. void show\_B() {
18. cout << "B = " << value << "\n";
19. };
21. void setVal(int val) {
22. value = val;
23. };
25. ~B() {
26. cout << "Работает деструктор B\n";
27. };
28. };
30. class D1 : virtual public B {
31. int value;
33. public:
34. D1(int v, int b)
35. : value(v) {
36. setVal(b);
37. cout << "Работает конструктор D1: параметр = " << value << "\n";
38. };
40. void show\_D1() {
41. cout << "D1 = " << value << "\n"; show\_B();
42. };
44. ~D1() {
45. cout << "Работает деструктор D1\n";
46. };
47. };
49. class D2 : virtual private B {
50. int value;
52. public:
53. D2(int v, int b)
54. :value(v) {
55. setVal(b);
56. cout << "Работает конструктор D2: параметр = " << value << "\n";
57. };
59. void show\_D2() {
60. cout << "D2 = " << value << "\n"; show\_B();
61. };
63. ~D2() {
64. cout << "Работает деструктор D2\n";
65. };
66. };
68. class D3 : virtual private B {
69. int value;
71. public:
72. D3(int v, int b)
73. :value(v) {
74. setVal(b);
75. cout << "Работает конструктор D3: параметр = " << value << "\n";
76. };
78. void show\_D3() {
79. cout << "D3 = " << value << "\n"; show\_B();
80. };
82. ~D3() {
83. cout << "Работает деструктор D3\n";
84. };
85. };
87. class D4 : private D1 {
88. int value;
90. public:
91. D4(int v, int d1, int b)
92. : D1(d1, b), value(v) {
93. cout << "Работает конструктор D4: параметр = " << value << "\n";
94. };
96. void show\_D4() {
97. cout << "D4 = " << value << "\n"; show\_D1();
98. };
100. ~D4() {
101. cout << "Работает деструктор D4\n";
102. };
103. };
105. class D5 : public D1, public D2, private D3 {
106. int value;
108. public:
109. D5(int v, int d1, int d2, int d3, int b)
110. :D1(d1, b), D2(d2, b), D3(d3, b), value(v) {
111. cout << "Работает конструктор D5: параметр = " << value << "\n";
112. };
114. void show\_D5() {
115. cout << "D5 = " << value << "\n"; show\_D1(); show\_D2(); show\_D3();
116. };
118. ~D5() {
119. cout << "Работает деструктор D5\n";
120. };
121. };
123. int main() {
124. system("color f0");
125. setlocale(LC\_ALL, "rus");
127. cout << "D4 temp(100, 200, 1);\n";
128. cout << "\nСледуя иерархии класса D4: \n";
129. D4 temp4(400, 100, 1);
130. temp4.show\_D4();
132. cout << "\n\nD5 temp(500, 100, 200, 300, 1);\n";
133. cout << "\nСледуя иерархии класса D5\n";
134. D5 temp5(500, 100, 200, 300, 1);
135. temp5.show\_D5();
137. cout << "\n\n";
138. }

**Результаты работы программы:**



**Вывод:**

В ходе лабораторной работы я получил практические навыки при использовании множественного наследования в языке С++.