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 лабораторным работам №18.1, 18.2**

Дисциплина: «Информатика»

Темы: Классы и объекты. Инкапсуляция, использование конструкторов.

Вариант 13

Выполнил:

Студент группы ИВТ-20-2б

Сафронов Владислав Владиславович

Проверила:

Доцент кафедры ИТАС

Полякова О.А.

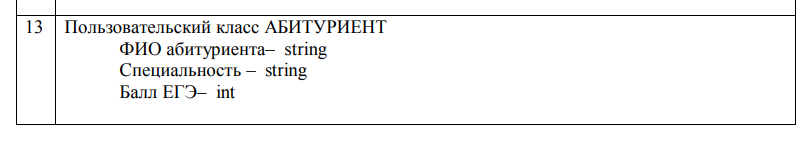
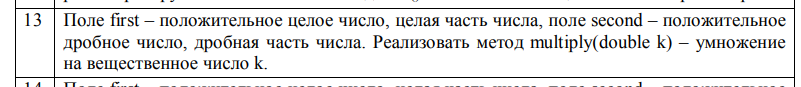
**Пермь, 2021**

**Цель задачи**

1. Создание консольного приложения, состоящего из нескольких файлов в системе программирования Visual Studio
2. Использование классов и объектов в ОО программе.
3. Создание объектов с использованием констукторов.

**Постановка задачи**

Реализовать определение нового класса.



**Анализ задачи**

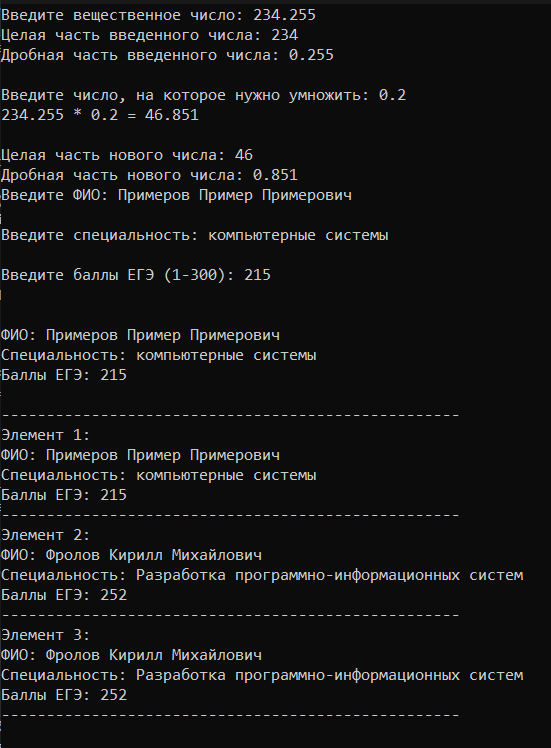
1. Для решения задачи необходимо:
   1. Разработать класс Fraction для хранения целой и вещественной пары числа и организовать методы для ввода числа, а также метод multiply для умножения полей на вещественное число k
   2. Разработать пользовательский класс Enrollee для хранения данных об абитуриенте: ФИО, специальность и количество баллов ЕГЭ, и организовать метод для ввода данных в поля, конструкторы, деструктор.
2. В ходе работы были использованы следующие типы данных:
   1. Класс Fraction для хранения целой и вещественной части числа: first и second
3. class Fraction
4. {
5. public:
6. int first;
7. double second;
   * 1. Методы init для ввода вещественного числа, разделения на целую и дробную части и последующий вывод частей в консоль. Можно передать в параметр вещественное число.
8. void init()
9. {
10. double tmp;
11. cout << "Введите вещественное число: ";
12. cin >> second;
13. second = modf(second, &tmp);
14. first = int(tmp);
15. cout << "Целая часть введенного числа: " << first << endl;
16. cout << "Дробная часть введенного числа: " << second << endl;
17. }
18. void init(double n)
19. {
20. double tmp;
21. second = modf(n, &tmp);
22. first = int(tmp);
23. cout << "Целая часть нового числа: " << first << endl;
24. cout << "Дробная часть нового числа: " << second << endl;
25. }
    * 1. Метод multiply для умножения вещественных числа из полей класса на вещественное число из входного параметра. Метод возвращает вещественное число - результат умножения.
26. double multiply(double n)
27. {
28. double f, s, t;
29. modf(first, &f);
30. s = fabs(modf(second, &t));
31. return ((s + f) \* n);
32. }
    1. Класс Enrollee для хранения данных об абитуриенте. Имеет 3 поля: строковое поле name для хранения ФИО абитуриента, строковое поле speciality для хранения специальности и целочисленное поле exams для хранения баллов ЕГЭ.
33. class Enrollee
34. {
35. string name;
36. string speciality;
37. int exams;
    * 1. Контрукторы класса: без параметров, с параметрами, а также конструктор копирования.
38. Enrollee() // без аргументов
39. {
40. name = "";
41. speciality = "";
42. exams = 0;
43. }
45. Enrollee(string a, string b, int c) // с аргументами
46. {
47. this->name = a;
48. this->speciality = b;
49. this->exams = c;
50. }
52. Enrollee(const Enrollee& obj) // копия
53. {
54. name = obj.name;
55. speciality = obj.speciality;
56. exams = obj.exams;
57. }
    * 1. Описывать деструктор явно не требуется, т.к. в классе не используется динамическая память.
58. ~Enrollee() {}
    * 1. Метод init для ввода полей с консоли.
59. void init()
60. {
61. string a;
62. int n;
64. cout << "Введите ФИО: ";
65. getline(cin, a);
66. while (a == "" || a[0] == ' ')
67. {
68. cout << "\nФИО введено некорректно.\n";
69. cout << "Введите ФИО: ";
70. getline(cin, a);
71. }
72. this->name = a;
73. cout << "\nВведите специальность: ";
74. getline(cin, a);
75. while (a == "" || a[0] == ' ')
76. {
77. cout << "\nСпециальность введена некорректно.\n";
78. cout << "Введите специальность: ";
79. getline(cin, a);
80. }
81. this->speciality = a;
82. cout << "\nВведите баллы ЕГЭ (1-300): ";
83. cin >> n;
84. while (n <= 0 || n > 300)
85. {
86. cout << "\nБаллы ЕГЭ введены некорректно.\n";
87. cout << "Введите баллы ЕГЭ (1-300: ";
88. cin >> n;
89. }
90. this->exams = n;
91. }
    * 1. Методы-гетторы GetName, GetSpeciality и GetExams, которые возвращают соответствующие поля класса.
92. string GetName() { return name; }
93. string GetSpeciality() { return speciality; }
94. int GetExams() { return exams; }
    * 1. Метод Print, который выводит поля класса в консоль.
95. void Print()
96. {
97. cout << "ФИО: " << this->name << endl;
98. cout << "Специальность: " << this->speciality << endl;
99. cout << "Баллы ЕГЭ: " << this->exams << endl;
100. cout << "---------------------------------------------------\n";
101. }
102. Поставленные задачи будут решены следующими действиями:
     1. В функции main() создается экземпляр класса Fraction и вызывается метод для ввода в него числа. Затем пользователь вводит число, на которое нужно умножить и вызывается метод multiply для умножения числа, в качестве входного параметра передается введенное число.
103. int main()
104. {
105. setlocale(LC\_ALL, "Russian");
106. system("chcp 1251>nul");
107. double n, res;
108. Fraction number;
109. number.init();
111. cout << "\nВведите число, на которое нужно умножить: ";
112. cin >> n;
113. res = number.multiply(n);
114. cout << (int)number.first + number.second << " \* " << n << " = " << res << endl << endl;
115. number.init(res);
116. cin.get();
     1. С помощью различных конструкторов создаются экземпляры класса Enrollee: obj1 – через конструктор без параметров, obj2 – через конструктор с параметрами, obj3 – через конструктор копирования. Затем для obj1 вызывается метод init для ввода данных в поля экземпляра класса. Результат ввода выводится в консоль. Затем создается вектор элементов класса Enrollee из уже инициализированных элементов и содержимое вектора в цикле выводится в консоль.
117. Enrollee obj1, obj2("Фролов Кирилл Михайлович", "Разработка программно-информационных систем", 252);
118. Enrollee obj3(obj2);
119. obj1.init();
121. cout << "\nФИО: " << obj1.GetName() << endl;
122. cout << "Специальность: " << obj1.GetSpeciality() << endl;
123. cout << "Баллы ЕГЭ: " << obj1.GetExams() << endl;
125. vector<Enrollee> lst{ obj1, obj2, obj3 };
127. cout << "\n---------------------------------------------------\n";
128. for (int i = 0; i < lst.size(); i++)
129. {
130. cout << "Элемент " << i + 1 << ":\n";
131. lst[i].Print();
132. }
133. }

**Блок-схема программы**

**Код программы**

1. #include <iostream>
2. #include <cmath>
3. #include <string>
4. #include <vector>
5. using namespace std;
7. class Fraction
8. {
9. public:
10. int first;
11. double second;
13. void init()
14. {
15. double tmp;
16. cout << "Введите вещественное число: ";
17. cin >> second;
18. second = modf(second, &tmp);
19. first = int(tmp);
20. cout << "Целая часть введенного числа: " << first << endl;
21. cout << "Дробная часть введенного числа: " << second << endl;
22. }
23. void init(double n)
24. {
25. double tmp;
26. second = modf(n, &tmp);
27. first = int(tmp);
28. cout << "Целая часть нового числа: " << first << endl;
29. cout << "Дробная часть нового числа: " << second << endl;
30. }
32. double multiply(double n)
33. {
34. double f, s, t;
35. modf(first, &f);
36. s = fabs(modf(second, &t));
37. return ((s + f) \* n);
38. }
39. };
41. class Enrollee
42. {
43. string name;
44. string speciality;
45. int exams;
46. public:
47. Enrollee() // без аргументов
48. {
49. name = "";
50. speciality = "";
51. exams = 0;
52. }
54. Enrollee(string a, string b, int c) // с аргументами
55. {
56. this->name = a;
57. this->speciality = b;
58. this->exams = c;
59. }
61. Enrollee(const Enrollee& obj) // копия
62. {
63. name = obj.name;
64. speciality = obj.speciality;
65. exams = obj.exams;
66. }
68. ~Enrollee() {}
70. void init()
71. {
72. string a;
73. int n;
75. cout << "Введите ФИО: ";
76. getline(cin, a);
77. while (a == "" || a[0] == ' ')
78. {
79. cout << "\nФИО введено некорректно.\n";
80. cout << "Введите ФИО: ";
81. getline(cin, a);
82. }
83. this->name = a;
84. cout << "\nВведите специальность: ";
85. getline(cin, a);
86. while (a == "" || a[0] == ' ')
87. {
88. cout << "\nСпециальность введена некорректно.\n";
89. cout << "Введите специальность: ";
90. getline(cin, a);
91. }
92. this->speciality = a;
93. cout << "\nВведите баллы ЕГЭ (1-300): ";
94. cin >> n;
95. while (n <= 0 || n > 300)
96. {
97. cout << "\nБаллы ЕГЭ введены некорректно.\n";
98. cout << "Введите баллы ЕГЭ (1-300: ";
99. cin >> n;
100. }
101. this->exams = n;
102. }
104. string GetName() { return name; }
105. string GetSpeciality() { return speciality; }
106. int GetExams() { return exams; }
108. void Print()
109. {
110. cout << "ФИО: " << this->name << endl;
111. cout << "Специальность: " << this->speciality << endl;
112. cout << "Баллы ЕГЭ: " << this->exams << endl;
113. cout << "---------------------------------------------------\n";
114. }
115. };
117. int main()
118. {
119. setlocale(LC\_ALL, "Russian");
120. system("chcp 1251>nul");
121. double n, res;
122. Fraction number;
123. number.init();
125. cout << "\nВведите число, на которое нужно умножить: ";
126. cin >> n;
127. res = number.multiply(n);
128. cout << (int)number.first + number.second << " \* " << n << " = " << res << endl << endl;
129. number.init(res);
130. cin.get();
132. Enrollee obj1, obj2("Фролов Кирилл Михайлович", "Разработка программно-информационных систем", 252);
133. Enrollee obj3(obj2);
134. obj1.init();
136. cout << "\nФИО: " << obj1.GetName() << endl;
137. cout << "Специальность: " << obj1.GetSpeciality() << endl;
138. cout << "Баллы ЕГЭ: " << obj1.GetExams() << endl;
140. vector<Enrollee> lst{ obj1, obj2, obj3 };
142. cout << "\n---------------------------------------------------\n";
143. for (int i = 0; i < lst.size(); i++)
144. {
145. cout << "Элемент " << i + 1 << ":\n";
146. lst[i].Print();
147. }
148. }

**Скриншоты результатов работы программы**

****

**Ответы на вопросы**

1. Что такое класс?

Класс – абстрактный тип данных, определяемый пользователем, представляет собой модель реального объекта в виде данных и функций для работы с ними.

1. Что такое объект (экземпляр) класса?

Переменная класса называется экземпляром класса или объектом. Класс объявляется один раз, а переменных такого класса может быть сколько угодно. Объявляются такие переменные так же как и переменные встроенных типов.

1. Как называются поля класса?

Полями/атрибутами называются данные класса.

1. Как называются функции класса?

Функции класса называются методами.



Спецификаторы доступа управляют видимостью элементов класса.



Спецификатор public используется, чтобы задать общедоступную видимость.



Спецификатор private используется, чтобы задать видимость только внутри класса для полей или методов под этим спецификатором.



Для спецификатора class по умолчанию используется private.



Для спецификатора struct по умолчанию используется public.



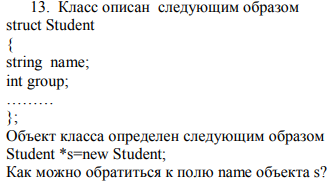
Интерфейс класса описывается после спецификатора public, чтобы к методам класса можно было обратится из других функций.



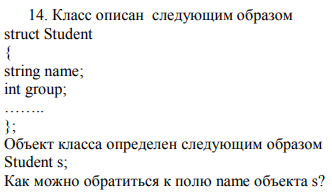
Значения атрибутов экземпляра класса можно изменить с помощью методов класса: модификаторов.



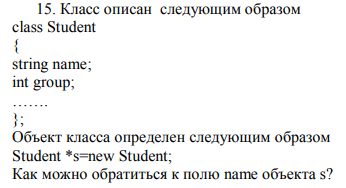
Значения атрибутов экземпляра класса можно получить с помощью методов класса: селекторов.



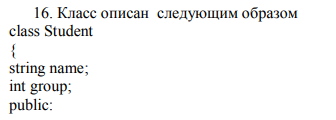
Через «->» можно обратиться к полю name, так как оно имеет открытый доступ public.

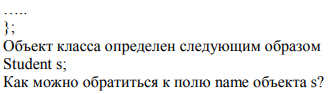


Через «.» можно обратиться к полю name, так как оно имеет открытый доступ public.

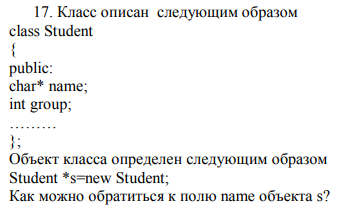


Необходимо прописать модификатор для класса Student.





Необходимо прописать модификатор для класса Student.

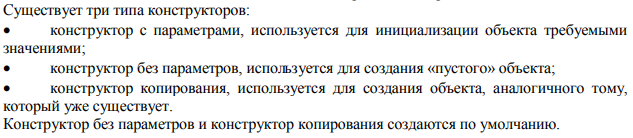


Необходимо прописать селектор и модификатор для класса Student.



Конструктор необходим для инициализации объекта.





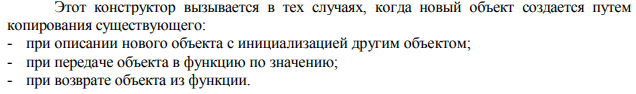


Деструктор – это особый вид метода, применяющийся для освобождения ресурсов, выделенных конструктором объекту. Описывать деструктор явным образом требуется в случае, когда объект содержит указатели на память, выделяемую динамически.

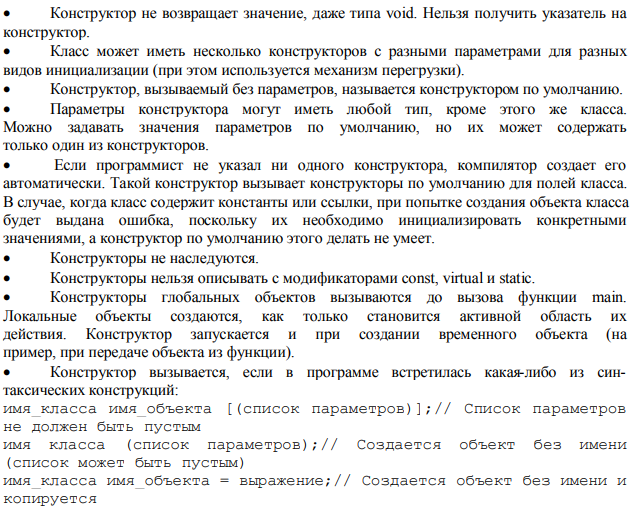


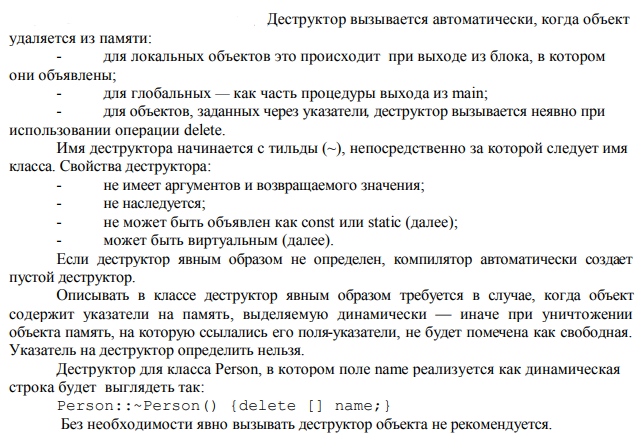
Конструктор с параметрами используется для инициализации объекта требуемыми значениями. Конструктор без параметров используется для создания «пустого» объекта. Конструктор копирования – это специальный вид конструктора, получающий в качестве единственного параметра указатель на объект этого же класса.





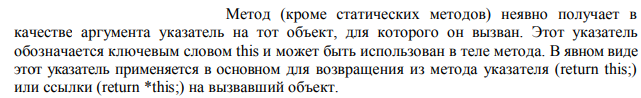


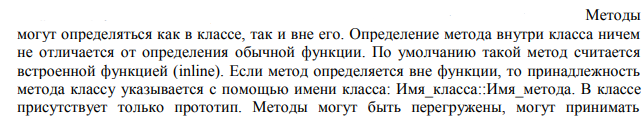












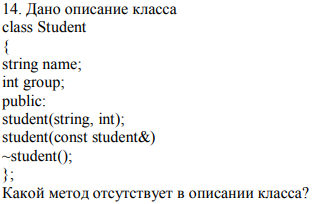
Конструктор не возвращает значения.



Конструкторы и деструкторы без параметров, если таковые не были прописаны явно.



Деструктор не возвращает значения.



Отсутствует конструктор без параметров в описании класса.



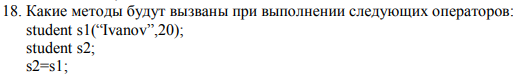
Будет вызван конструктор без параметров.



Будет вызван конструктор с параметрами.



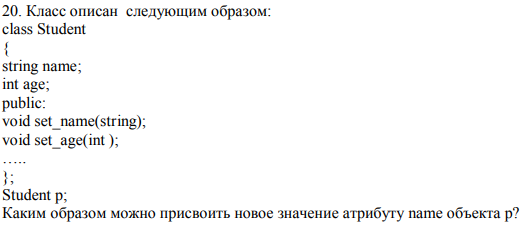
Будут вызваны конструктор с параметрами, а затем конструктор копирования.



Будут вызваны конструктор с параметрами, затем конструктор без параметров, а затем конструктор копирования.



Конструктор без параметров.



C помощью метода set\_name().