МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

ИНСТИТУТ НЕПРЕРЫВНОГО И ДИСТАНЦИОННОГО ОБРАЗОВАНИЯ

|  |
| --- |
| КАФЕДРА КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ И ПРОГРАММНОЙ ИНЖЕНЕРИИ |

ОЦЕНКА

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| старший преподаватель |  |  |  | Е.О. Шумова |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №2 |
| Классы, конструкторы, деструкторы |
| по дисциплине: Объектно-ориентированное программирование |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ гр. № | Z1431 |  |  |  | М.Д. Быстров |
|  | номер группы |  | подпись, дата |  | инициалы, фамилия |
| Студенческий билет № | 2021/3572 | |  |  |  |

Санкт-Петербург 2023

**Условие**

Цель работы: изучить принципы создания классов с конструкторами, применив на практике знания базовых синтаксических конструкций языка C++ и объектно-ориентированного программирования.

Закрепить знания по теме: классы, конструкторы, деструкторы.

Описание работы: в работе необходимо реализовать класс в соответствии с вариантом задания и создать приложение. В классе должны быть предусмотрены: - конструктор для установки начальных значений полей, -конструктор по умолчанию, -конструктор копирования, деструктор. Поля класса должны иметь спецификатор доступа private. Доступ к полям осуществляется через public методы. В функции main() создается не менее 3 объектов класса (с использованием всех конструкторов) и осуществляется вызов методов класса.

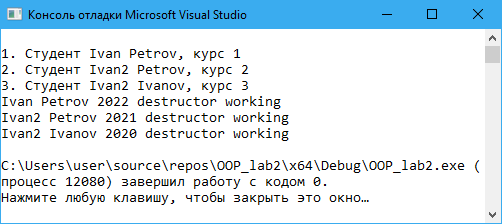
Индивидуальное задание (вариант 2):

2. Определите класс Student (Студент). Этот класс должен иметь следующие поля: name (имя), surname (фамилия), year (год поступления в вуз). Класс должен иметь метод getFullName(), с помощью которого можно вывести одновременно имя и фамилию студента. Также класс должен иметь метод getCourse(), который будет выводить текущий курс студента (от 1 до 5). Курс вычисляется так: нужно от текущего года отнять год поступления в вуз. Текущий год получите самостоятельно.

**Полный текст (листинг) программы**

1. Файл «main.cpp»
2. #include <string>
3. #include <iostream>
4. #include <Windows.h>
5. #include "Student.h"
7. using namespace std;
9. int main()
10. {
11. **SetConsoleCP(1251);**
12. SetConsoleOutputCP(1251);
14. cout << "ЛР №2 ВАРИАНТ 2" << endl << endl;
16. **Student\* student1 = new Student();**
18. student1->setName("Ivan");
19. student1->setSurname("Petrov");
20. student1->setYear(2022);
22. Student\* student2 = new Student(\*student1);
24. student2->setName("Ivan2");
25. student2->setYear(student1->getYear() - 1);
27. Student\* student3 = new Student(
28. student2->getName(),
29. "Ivanov",
30. student2->getYear() - 1);
32. Student\*\* students = new Student\*[3];
34. students[0] = student1;
35. students[1] = student2;
36. **students[2] = student3;**
38. for (int i = 0; i < 3; i ++)
39. {
40. cout << (i + 1) << ". Студент " << students[i]->getFullName()
41. **<< ", курс " << students[i]->getCourse() << endl;**
42. }
44. delete student1;
45. delete student2;
46. **delete student3;**
47. delete[] students;
49. return 0;
50. }
51. Файл «Student.h»
52. #include <string>
54. using namespace std;
56. **class Student**
57. {
58. private:
60. string name;
61. **string surname;**
62. int year;
64. public:
66. **Student();**
67. Student(string name, string surname, int year);
68. Student(const Student &student);
69. ~Student();
71. **string getName();**
72. string getSurname();
73. int getYear();
75. void setName(string name);
76. **void setSurname(string surname);**
77. void setYear(int year);
79. string getFullName();
80. int getCourse();
81. **};**
82. Файл «Student.cpp»
83. #include <ctime>
84. #include <string>
85. #include <iostream>
86. #include "Student.h"
88. Student::Student()
89. {
91. }
93. Student::Student(string name, string surname, int year)
94. {
95. this->name = name;
96. this->surname = surname;
97. **this->year = year;**
98. }
100. Student::Student(const Student &student)
101. {
102. **this->name = student.name;**
103. this->surname = student.surname;
104. this->year = student.year;
105. }
107. **Student::~Student()**
108. {
109. std::cout << name << " " << surname << " " << year
110. << " destructor working" << std::endl;
111. }
113. string Student::getName()
114. {
115. return this->name;
116. }
118. string Student::getSurname()
119. {
120. return this->surname;
121. }
123. int Student::getYear()
124. {
125. return this->year;
126. }
128. void Student::setName(string name)
129. {
130. this->name = name;
131. }
133. void Student::setSurname(string surname)
134. {
135. this->surname = surname;
136. }
138. void Student::setYear(int year)
139. {
140. this->year = year;
141. }
143. int Student::getCourse()
144. {
145. time\_t t = time(nullptr);
146. tm\* now = new tm;
148. localtime\_s(now, &t);
150. int currentYear = now->tm\_year + 1900;
152. **if (currentYear >= this->year)**
153. {
154. return currentYear - this->year;
155. }
156. else
157. **{**
158. return 0;
159. }
160. }
162. **string Student::getFullName()**
163. {
164. return this->name + " " + this->surname;
165. }

**Работа программы**



*Рисунок 1 Работа программы*

**Выводы**

В ходе выполнения лабораторной работы №2 были получены навыки по созданию классов, созданию и перегрузке конструкторов, созданию деструкторов.

Созданы конструкторы: по умолчанию, конструктор копирования, конструктор с инициализацией начальных значений полей. Продемонстрировано создание объектов с использованием каждого конструктора.

Создан деструктор. В консоли продемонстрирована работа деструктора.

Получены навыки по работе со временем с помощью стандартной библиотеки C++.