Обектно-ориентирано програмиране 2024/2025, спец. Софтуерно инженерство
Контролна работа No. 2

1. Реализирайте на C++ клас Vector2D, описващ обобщен вектор в евклидова равнина. Реализирайте и:

- конструктор по подразбиране, инициализиращ го като нулев вектор
- комутативен оператор * реализиращ външна приятелска функция за умножение на вектора с число double (като първи аргумент)
- оператори реализиращи сума на два вектора (оператори + и +=)
- оператор ^ реализиращ скаларно произведение на вектора с друг подаден вектор по координатно
- оператори == и !=

2. Да се реализира следната полиморфна йерархия на C++:
Клас връх (Peak), който има височина и име на съдържащата го планина.
Клас туристически връх (TPeak), който наследява връх (Peak) и добавя името на върха.
Клас военен връх (MPeak), който наследява връх (Peak) и добавя номер на котата на върха.

За класовете реализирайте подходящи конструктори, методи за достъп и метод за извеждане на името на върха (за класа MPeak, името е във формат “кота 54291”).

3. Обяснете кратко как работи множественото наследяване в C++. Как виртуалните функции и полиморфизмът се използват в контекста на множествено наследяване?

- Клас Report трябва да наследни Printable и Loggable. Допишете го.
- Използвайте полиморфизъм за print() и log(), така че на ред 41 и ред 42 да се вика Report::print и Report::log. Коригирайте или допишете кода, ако се налага.
- C++ какво свързване ще използва след вашите промени?
- Какво ще отпечата кодът по-долу преди и след вашите промени. Обосновете отговора си.

```
1  #include <iostream>
2  using namespace std;
3
4  class Printable {
5  public:
6      void print() const {
7          cout << "Printable base class" << endl;
8      }
9  };
10
11 class Loggable {
12 public:
13     void log() const {
14         cout << "Loggable base class" << endl;
15     }
16 };
17
18
19 class Report {
20 public:
21     void print() const {
22         cout << "Report is being printed." << endl;
23     }
24
25     void log() const {
26         cout << "Report log entry created." << endl;
27     }
28 };
29
30 void processPrintable(const Printable* p) {
31     p->print();
32 }
33
34 void processLoggable(const Loggable* l) {
35     l->log();
36 }
37
38 int main() {
39     Report myReport;
40
41     processPrintable(&myReport);
42     processLoggable(&myReport);
43
44     return 0;
45 }
46
```