Практически изпит по ООП за 3

Вариант А

UNIX времево клеймо (timestamp), още наричано "epoch time", е цяло число, указващо брой секунди изминали след 1 януари 1970 г. UTC. Такова клеймо позволява всеки момент във времето след 01.01.1970 г. да се съхранява като едно-единствено цяло число. Ако едно клеймо A е по-малко от друго клеймо B, можем да заключим, че A е момент от времето, който е преди B.

Да се реализира клас Timestamp, представящ UNIX времево клеймо. Класът трябва да спазва принципа за енкапсулация и да няма външен достъп до член-променливите на класа. В класа трябва да се включат:

- променлива от тип unsigned long, която пази клеймото;
- конструктор по подразбиране, който задава на клеймото стойност нула (съответстващ на дата 01.01.1970 г.)
- конструктор, приемащ един параметър – стойност за клеймото, с която да се инициализира класа;
- функции GetValue и SetValue, които позволяват съответно да се достъпва и променя клеймото, съхранено в класа;

За класа да се предефинират следните оператори:

- operator << за изход в поток;
- operator >> за вход от поток.

- operator+= за добавяне на зададен брой секунди към клеймото. Операторът получава цяло число (брой секунди) и ги добавя към клеймото в класа.

- operator+ за пресмятане на ново клеймо, което се получава с добавяне на даден брой секунди към дадено клеймо. Операторът получава брой секунди d, прибавя ги към клеймото на обекта, за който е извикан и връща нов обект, съдържащ новото клеймо.

Да се дефинира нов клас TimestampWithDescription, производен на класа Timestamp. Този клас трябва да надгради Timestamp като добави възможност да се задава описание на клеймото. За целта в този клас трябва да може да се съхранява символен низ с произволна дължина, който ще съдържа описанието. Класът трябва да има:

- всички функции от голямата четворка;
- функции GetDescription и SetDescription, които позволяват съответно да се достъпва и променя описанието;
- operator << за изход в поток;
- operator >> за вход от поток.

В main функцията на програмата да се реализира код, който отваря текстов файл с име timestamps.txt, в който на всеки ред е дадено едно клеймо и неговото описание. Например:

12345 Първо клеймо

923498 Друго клеймо

12 Тест Тест Тест Тест

Програмата трябва да обработи цялата информация от файла по следния начин: данните от всеки ред се прочитат, създава се TimestampWithDescription обект, в него се записват прочетените данни, след което обектът се извежда на екрана.

Всички символни низове в програмата трябва да се представят динамично, т.е. За всеки от тях трябва да се заделя памет с new, да се освобождава с delete и да има указател char*, който сочи към нея. НЕ Е ПОЗВОЛЕНО да се използват статични масиви(например char buffer[100]), НИТО класът std::string.

Вариант Б

Да се реализира йерархия от класове, които описват печатни издания.

Базов клас за йерархията да бъде Печатно издание (Publication). Всяко такова трябва да има заглавие (символен низ с произволна дължина) и тираж (цяло число без знак).

Наследници на класа трябва да бъдат книга (Book), която има и автор (символен низ с произволна дължина) и списание (Magazine), което има номер на брой (цяло неотрицателно число). НЕ е позволено използването на статични низове, НИТО на класа std::string.

За всеки от класовете да се реализират:

- Голямата четворка, където е нужна
- Подходящи Get/Set функции за член-данните на класовете
- operator << за изход в поток;
- operator >> за вход от поток.

Използването на йерархията да се демонстрира в програма, която:

- Въвежда от стандартния вход десет книги и десет списания
- Извежда в текстов файл popular.txt онези книги и списания, които имат тираж от поне хиляда броя. При извеждането всяко издание да бъде на отделен ред във файла, като за книгите този ред съдържа заглавие, тираж, автор, а за списанията — заглавие, тираж, брой.

Всички символни низове в програмата трябва да се представят динамично, т.е. За всеки от тях трябва да се заделя памет с new, да се освобождава с delete и да има указател char*, който сочи към нея. НЕ Е ПОЗВОЛЕНО да се използват статични масиви (например char buffer[100]), НИТО класът std::string.

Вариант B

1. Да се създаде клас Student, описващ студент със следните атрибути:

Име (символен низ с големина до 4 MB)
Факултетен номер (естествено число с 6 цифри)
Оценка

Класът да реализира принципа на капсулацията и да поддържа средства за достъп до информацията за студента, както и каноничните методи (голямата четворка), ако са необходими.

2. За клас Student да се реализира сериализация (оператори за еднозначен изход в поток и вход от поток).

3. Във файла “students1.txt” се съдържа информация за студенти във формата на сериализацията от предишното условие.
Някои от студентите във файла може да се повтарят. Приемаме, че факултетният номер е уникален идентификатор на студентите (не е възможно двама студента с еднакъв ФН да имат различни имена).

Във файла “students2.txt” да се изведе информация за студентите от файла “students1.txt”, където са премахнати повторенията на студенти.

Ако за даден студент X в “students1.txt” има повече от един запис, оценката на X, отпечатана в “students2.txt”, да е средно аритметично
на всички оценки на X в “students1.txt”.

Вариант Г

1. Да се създаде клас Product, описващ продукт със следните атрибути:

Име (символен низ с големина до 4 MB)
Инвентарен номер (естествено число с 6 цифри)
Цена

Класът да реализира принципа на капсулацията и да поддържа средства за достъп до информацията за продукта, както и каноничните методи (голямата четворка), ако са необходими.

2. За клас Product да се реализира сериализация (оператори за еднозначен изход в поток и вход от поток).

3. Във файла “products1.txt” се съдържа информация за продукти във формата на сериализацията от предишното условие.
Някои от продуктите във файла може да се повтарят. Приемаме, че инвентарният номер е уникален идентификатор на продуктите (не е възможно два продукта с еднакъв инвентарен номер да имат различни имена).

Във файла “products2.txt” да се изведе информация за продуктите от файла “products1.txt”, където са премахнати повторенията на продукти.

Ако за даден продукт X в “products1.txt” има повече от един запис, цената на X, отпечатана в “products2.txt”, да е най-ниската от всички
цени на X в “products1.txt”.

Вариант Д

Да се реализира йерархия от класове, която описва средства за писане. Базов клас за йерархията да бъде Пишещо средство (WritingInstrument). Всяко такова трябва да има марка (символен низ с произволна дължина) и година на производство (цяло неотрицателно число).

Наследници на класа трябва да бъдат Маркер (Marker), който има и цвят (символен низ с произволна дължина) и Молив (Pencil), който има твърдост (цяло число между 0 и 14).
За всеки от класовете да се реализират:

- Голямата четворка (там където е нужна);
- Подходящи Get/Set функции за член-данните на класовете;
- operator<< за изход в поток;
- operator>> за вход от поток.

Използването на йерархията да се демонстрира в програма, която:

- Въвежда от стандартния вход пет маркера и пет молива и след това символен низ, задаващ марка;
- Извежда в текстов файл branded.txt информация за онези от тях, които са от въведената марка. При извеждането всяко пишещо средство да бъде на отделен ред във файла, като за маркерите този ред съдържа марка, година на производство и цвят, а за моливите — марка, година на производство и твърдост.

Всички символни низове в програмата трябва да се представят динамично, т.е. За всеки от тях трябва да се заделя памет с new, да се освобождава с delete и да има указател char*, който сочи към нея. НЕ Е ПОЗВОЛЕНО да се използват статични масиви (например char buffer[100]), НИТО класът std::string.

Вариант E

UNIX времево клеймо (timestamp), още наричано "epoch time", е цяло число, указващо брой секунди изминали след 1 януари 1970 г. UTC. Такова клеймо позволява всеки момент във времето след 01.01.1970 г. да се съхранява като едно-единствено цяло число. Ако едно клеймо A е по-малко от друго клеймо B, можем да заключим, че A е момент от времето, който е преди B.

Да се реализира клас Timestamp, представящ UNIX времево клеймо. Класът трябва да спазва принципа за енкапсулация и да няма външен достъп до член-променливите на класа. В класа трябва да се включат:

- променлива от тип unsigned long, която пази клеймото;
- конструктор по подразбиране, който задава на клеймото стойност текущото системно време (стойността, която връща стандартната функция time() от библиотеката <ctime>)
- конструктор, приемащ един параметър — стойност за клеймото, с която да се инициализира класа;
- функции GetValue и SetValue, които позволяват съответно да се достъпва и променя клеймото, съхранено в класа;

За класа да се предефинира пълен комплект оператори за наредба (шест функции за операторите <, >, <=, >=, == и !=).

Да се реализира клас Appointment, който съдържа обект от класа Timestamp. Класът Appointment трябва да добавя възможност да се задава описание на дадена уговорка. За целта в този клас трябва да може да се съхранява символен низ, който ще съдържа описанието. За класа да се реализират:

- всички функции от голямата четворка;
- функции GetDescription и SetDescription, които позволяват съответно да се достъпва и променя описанието;
- operator << за изход в поток;
- operator >> за вход от поток.

В main функцията на програмата да се реализира код, който отваря текстов файл с име calendar.txt, за който на всеки ред е дадено текстовото описание на една уговорка.

Пример:

12345 Роден е бай Петко
923498 Да се видя с Гошо
12 Тест Тест Тест Тест

Програмата трябва да обработи цялата информация от файла по следния начин: данните от всеки ред се прочитат, създава се Appointment обект, в него се записват прочетените данни, след което обектът се извежда на екрана.

Всички символни низове в програмата трябва да се представят динамично, т.е. За всеки от тях трябва да се заделя памет с new, да се освобождава с delete и да има указател char*, който сочи към нея. НЕ Е ПОЗВОЛЕНО да се използват статични масиви (например char buffer[100]), НИТО класът std::string.