Домашнее задание 3
Классы. Примеры.
19.02.2021

Вам необходимо смоделировать телефонную записную книжку. В книжку можно записывать номера, выводить все контакты, совершать звонки по шаблону номера или имени и выводить наиболее популярный контакт.

Класс «контакт»:

```
class Contact{
    string name;       //имя человека
    int prefix;        //префикс номера (число из 3 цифр) 000-333
    int number;        //номер телефона (число из 7 цифр) 0000000-9999999
    int calls;         //количество звонков на этот номер
public:
    Contact();
    void setContact(const string & newName, int newPrefix, int newNumber);
    bool nameLike(const string & partOfName);
    bool numberLike(int numberSuffix);
    int getCalls();
    void newCall();
    void print();
};
```

Необходимо описать:

1. `Contact()` — конструктор по умолчанию (инициализирует все поля);
2. `void setContact(const string & newName, int newPrefix, int newNumber)` — метод для задания (сеттер) имени абонента, префикса и номера (корректность проверять не нужно);
3. `bool nameLike(const string & partOfName);` — метод для проверки совпадения префикса имени абонента с данным параметром, возможно совпадение с целым именем (используйте `string str.substr (from, to)`); например, `A.nameLike("Si")` вернет `true`, если `(A.name == "Sid")`;
4. `bool numberLike(int endOfNumber);` — метод для проверки совпадения суффикса номер абонента с данным параметром, возможно совпадение с целым номером (лучше всего это реализовать в цикле с проверкой всех остатков при делении на 10^K); например, `A.numberLike(567)` вернет `true`, если `A.number == 1234567`;
5. int getCalls() — метод для подсчета количества звонков (параметр `calls`);
6. void newCall() — метод для отметки о новом звонке (параметр `calls`);
7. void print() — метод для печати строки результатов (имя абонента, префикс, номер, количество звонков). Для красивого вывода пользуйтесь манипулятором `std::setw()` из библиотеки `<iomanip>`;

Класс «записная книга»:

```
class PhoneBook{
    int abonentNumber;  //количество записей
    Contact names[10];      //записи
    void printHeader();
public:
    PhoneBook();
    void addAbonent(const string & name, int pref, int number);
```

```
    bool callAbonent(const string & prefixOfName);
    bool callAbonent(int suffixOfNumber);
    void print();
    void printTheMostCommon();
};
```

Необходимо описать:

1. void printHeader(); — метод для печати строки шапки (имя абонента, префикс, номер, количество звонков). Для красивого вывода пользуйтесь манипулятором `std::setw()` из библиотеки `<iomanip>`;

2. PhoneBook() — конструктор (инициализирует поле);

3. void addAbonent(const string & name, int pref, int number) — метод для добавления абонента. Если имя абонента содержит более 10 символов, номер содержит более 9 цифр или количество абонентов уже 10, то вывести сообщение о некорректности («FAIL: incorrect name or number format») и ничего не добавлять, иначе — вывести сообщение о корректности («ADD: name phone») добавить запись. Проверять есть ли уже такой абонент в книге НЕ НУЖНО. Вызывается при команде «+ name prefix number».

4. bool callAbonent(const string & prefixOfName) — метод для осуществления звонка по началу имени абонента. Вывести всех подходящих (графу call оставлять старой). Если подходящих абонентов больше или меньше одного, то сообщить об этом («choose unique» и «no such abonent» соответственно) и не звонить, иначе позвонить (`Contact::newCall()`) и сообщить об этом. Вызывается при команде «#1 name».

5. bool callAbonent(int suffixOfNumber) — метод для осуществления звонка по концу номера абонента. Вывести всех подходящих (графу call оставлять старой). Если подходящих абонентов больше или меньше одного, то сообщить об этом («FAIL: choose unique» и «FAIL: no such abonent» соответственно) и не звонить, иначе позвонить (`Contact::newCall()`) и сообщить об этом («CALL»). Вызывается при команде «#2 number».

6. void print() — метод для печати результатов всех команд с помощью метода void Team::print(). Вызывается при команде «?».

7. void printTheMostCommon() — метод для поиска и печати абонента, которому звонили больше всего. Если таких абонентов несколько, то печатать всех. Вызывается при команде «*».

Ввод	Вывод
`|+ Sid 777 1234567`	`| ADD: Sid 777-1234567`
`|+ Scrat 700 1111567`	`| ADD: Scrat 700-1111567`
`|+ Diego 1000 123`	`| FAIL: incorrect name or number format`
`|?`	`|name pref number calls`
`|`	`|Sid 777 1234567 0`
`|`	`|Scrat 700 1111567 0`
`|#2 4567`	`|name pref number calls`
`|`	`|Sid 777 1234567 0`
`|`	`| CALL`
`|#2 567`	`|name pref number calls`
`|`	`|Sid 777 1234567 1`
`|`	`|Scrat 700 1111567 0`
`|`	`| FAIL: choose unique`
`|#1 Sc`	`|name pref number calls`
`|`	`|Scrat 700 1111567 0`
`|`	`| CALL`
`|#1 Sid`	`|name pref number calls`
`|`	`|Sid 777 1234567 1`
`|`	`| CALL`
`|*`	`|name pref number calls`
`|`	`|Sid 777 1234567 2`
`|#1 Diego`	`|name pref number calls`
`|`	`| FAIL: no such abonent`
`|.`	`|`

1. Описать 2 класса: товар (`class Good`) и каталог (`class Catalog`). Товар описывается 4 параметрами: порядковый номер (`const int id`), название (`const string name`), цена (`const int price`) и количество товара на складе (`int number`). Каталог содержит количество различных товаров (`int Catalog::numberOfGoods`), которое передается в конструкторе, и сам массив указателей на товары (`class Catalog::Goods ** list`), который создается динамически в конструкторе (а очищается в деструкторе). Товары создаются тоже динамически. Порядковые номера товаров присваивает каталог (начиная с 0). Необходимо описать методы покупки (`void Catalog::buy(string name, int number)` и `void Catalog::buy(int id, int number)`), который проверяет, возможна ли покупка данного товара в указанном количестве (если покупка осуществилась успешно, то вычесть указанное количество товара со склада). После обработки всех запросов на покупки распечатать весь каталог (`void Catalog::print() const`).

   На вводе дано число n — количество товаров в каталоге. Далее каталог формируется n запросами вида `+ name price number`. Потом следуют число m — количество запросов на покупку. Далее запросы на покупку вида вида `? name` или `# id`.

   Замечание: класс каталог является другом к классу товар.

Ввод	3 + water 100 33 + juice 200 21 + banana 150 29 5 ? banana 10 # 0 15 # 2 40 # 5 10 ? banan 10
Вывод	banana x 10 = 1500 water x 10 = 1000 banana x 40 = not enough 5 = incorrect index banan = incorrect name # name price number 0 water 100 23 1 juice 200 21 2 banana 150 19

2. Описать 2 класса: герой (`class Hero`) и оружие (`class Weapon`). Герой описывается 4 параметрами: имя (`string`), атака (`int attack`), защита (`int protection`) и оружие (`Weapon weap`). Оружие описывается 2 параметрами: тип оружия (внутренний перечислимый тип `enum type{ATTACK, PROTECTION, NONE}` и сила оружия (`int value`). Атакующая сила героя — это сумма атаки героя и силы атакующего оружия (если оружие атакующего типа). Защитная сила героя — это сумма защиты героя и силы защитного оружия (если такое оружие есть). Герой A побеждает героя B если, атакующая сила героя A строго больше защитной силы героя B.

   Дано целое n — количество героев и количество единиц оружия. Далее вводится n имен героев (изначально ни у кого оружия нет, то есть тип `NONE`), их нападение и защиту. Потом следует целое m — количество единиц оружия. Далее вводится m строк, описывающих оружие (тип оружия, сила оружия и имя героя). Если у героя уже есть оружие и ему предлагают еще одно, то он выбирает оружие с большей силой. Для сравнения оружия опишите метод `bool Weapon::operator>(const Weapon &)`. Для сравнения героев опишите метод `bool Hero::operator>(const Hero&)`. Вывести кто кого побеждает (перебрать все пары различных героев).

   Замечание: нельзя использовать дружественные классы и методы (при необходимости описать методы `get() const`), массив героев сделать динамическим массивом в `int main()`.

Ввод	3 Dima 5 5 Kuat 6 1 Pasha 4 4 5 1 attack Pasha 4 protect Kuat 3 protect Pasha 2 attack Kuat 5 attack Pasha
Вывод	Dima win Pasha Kuat win Dima Kuat win Pasha Pasha win Dima Pasha win Kuat

   Пояснение к примеру: Дима без оружия (атака: 5, защита: 5), Куат с защитным оружием силы 4 (атака: 6, защита: 5), Паша с атакующим оружием силы 5 (атака: 9, защита: 4).

3. Описать класс `Pair`, содержащий 2 поля и конструкторы. От него отнаследовать 2 класса: рациональные числа и комплексные числа (с целыми значениями по отдельным компонентам). Определить операции сложения, вычитания и умножения и производных классов. Изначально имеется 1 рациональное число `0/1` и одно комплексное `0+1*i`. Необходимо провести с ними некоторые операции. У класса рациональных чисел определить функцию сокращения (делить числитель и знаменатель на НОД, который ищется алгоритмом Евклида). У класса `Pair` и наследников определить виртуальный метод `print()`.

   На вводе дано число n — количество запросов. Далее n запросов, которые преобразуют числа (сначала идет тип операции: 1 или 2; потом операция: +, -, *; потом число, определенное 2 числами).

Ввод	6 1 + 2 3 1 + 1 3 1 - 1 3 1 * 1 2 2 + 1 0 2 * 0 1
Вывод	2/3 1/1 2/3 1/3 1+1*i -1+1*i

4. Описать классы: четырехугольник `Quad`, параллелограмм (отнаследован от четырехугольника), квадрат (отнаследован от параллелограмма). Класс четырехугольник имеет поля: 4 стороны и виртуальные методы `void print()`.

   Дано n, далее n описаний сторон (4 числа). По вводимым данным определять тип четырехугольника (для этого определить статические методы `bool is(int, int, int, int)`) и создавать соответствующий объект динамически в массив указателей `Quad **ptr`. Далее распечатать все объекты (тип объекта печатается в соответствующих методах `void print()`).

Ввод	5 1 2 3 4 1 2 2 1 1 2 1 2 2 2 2 2 3 2 3 2
Вывод	Quad: 1 2 3 4 Quad: 1 2 2 1 Parallelogram: 1 2 Square: 2 Parallelogram: 3 2

5. Описать класс `class Computer` с параметрами частоты процессора `int clockRate` и объемом оперативной памяти `int RAM` и соответствующими `get`-функциями. Отнаследовать класс `class Notebook` с параметрами: длительность батареи `int Batery` и название производителя `string Vendor`. Отнаследовать класс `class Monoblock` с параметром: диагональ `int Display`.

   На вводе дано число n — количество аппаратов. Далее n описаний устройств:
   ? — неизвестное устройство,
   N — ноутбук,
   M — моноблок.
   Распечатать устройства с максимальной частой и максимальным объемом оперативной памяти.

Ввод	6 N 2000 8000 5 ASUS ? 3000 4000 M 2500 6000 25 M 1500 2000 21 N 2500 6000 7 HP ? 2300 8000
Вывод	Maximum clockRate: Monoblock 2500 GHz 6000 Mb 25" Notebook 2500 Ghz 6000 Mb 7 hour HP Maximum RAM: Notebook 2000 Ghz 8000 Mb 5 hour ASUS Computer 2300 Ghz 8000 Mb