Экзаменационные проекты по дисциплине
«Объектно-ориентированное программирование»
для студентов группы 335
Транспортно-технологического факультета
Сыктывкарского Лесного Института,
2022–2023 учебный год, 5 семестр

Макаров П. А., доцент кафедры ФиАТПиП

Сыктывкар, 2022

Содержание

1 Общие требования **1**
1.1 Предоставляемые материалы 1
1.2 Сроки выполнения 2

2 Предлагаемые проекты **2**

Список литературы **5**

1 Общие требования

Выполнение данных проектов не является обязательным, но даёт возможность сдать экзамен досрочно. Каждый студент должен выполнить свой индивидуальный проект самостоятельно. При желании, для проекта можно выбрать свою собственную тему, однако необходимо предварительно обсудить это с преподавателем.

1.1 Предоставляемые материалы

1. Архив со всеми исходными текстами.

2. Отчёт с подробным описанием хода выполнения проекта.

1.2 Сроки выполнения

К проверке принимаются работы, отправленные преподавателю тем или иным способом (электронной почтой, сообщениями в социальных сетях, ссылкой на облачное хранилище либо репозиторий) **до 23:59 12 января 2023 г.** Перед отправкой убедитесь, что работа выполнена должным образом, грамотно прокомментирована и снабжена понятным сопроводительным письмом.

2 Предлагаемые проекты

1. [10, №№ 10.08, 10.09, 10.14] Как известно, *рациональное число* — это несократимая дробь вида n/m, где $n \in \mathbb{Z}$, $m \in \mathbb{N}$. Опишите класс `Rational`, реализующий понятие рационального числа, воспользовавшись числами типа `long long` для представления числителя и знаменателя дроби. Снабдите конструкторы класса `Rational` проверкой, что знаменатель отличен от нуля. Если проверка не пройдена, выбрасывайте исключение, для которого разработайте специальный класс.

   (a) Требование о несократимости дроби на первом этапе проигнорируйте. Снабдите класс четырьмя основными действиями арифметики, соответствующими им операциями присваивания (`+=`, `-=`, `*=`, `/=`), функциями преобразования к числу типа `double` и к целому (реализуйте две функции: через отбрасывание дробной части и через округление к ближайшему). Предусмотрите методы для извлечения из объекта значений числителя и знаменателя.

   Напишите набор тестов, демонстрирующий корректность работы данной версии класса.

   (b) Возможности предыдущей версии класса несколько ограничены: все арифметические операции имеют тенденцию наращивать значение знаменателя — как напрямую при выполнении умножения и деления, так и через приведение к общему знаменателю, которое необходимо при сложениях и вычитаниях. Рано или поздно значение знаменателя превышает возможности разрядности даже для `long long`.

   Напишите тесты, демонстрирующие ограничения нехватки разрядности. Желательно при этом найти точные границы возможностей.

   Используя алгоритм Евклида, напишите функцию, определяющую наибольший общий делитель для двух чисел типа `long long`. Обратите внимание, что наибольший общий делитель для дух чисел всегда по определению положителен и не меняется при любой смене знаков исходных чисел. Поскольку эта функция потребуется вам в классе `Rational`, она должна быть его приватным методом, но так как объект `Rational` ей для работы не нужен, следует сделать её статической.

   Воспользовавшись этой функцией, усовершенствуйте ваши арифметические операции так, чтобы, во-первых, результат их работы всегда

представлял собой несократимую дробь; во-вторых, чтобы “лишние” множители по возможности изымались из чисел до выполнения других операций: так при перемножении двух дробей

$$\frac{n}{m} \cdot \frac{p}{q}$$

следует не только предварительно сократить обе дроби, если этого ещё не сделано, но и проверить, нет ли нетривиального (т. е. отличного от единицы) общего делителя в парах (n, q) и (m, p).

С помощью написанных ранее тестов убедитесь, что теперь возможности вашего класса шире. Напишите тесты, демонстрирующие границы возможностей новой реализации, и сравните эти границы со старыми.

2. [10, №10.11] Опишите абстрактный класс `Body`, представляющий некое абстрактное физическое тело, для которого не задана форма, но задана плотность составляющего его вещества. Плотность задаётся полем типа `double`. Предусмотрите в классе чисто виртуальную функцию `Volume()`, задающую объём тела, и простую функцию `Mass()`, вычисляющую массу как произведение объёма и плотности.

   Унаследуйте от класса `Body` класс `Cube`, представляющий куб, сделанный из однородного вещества и задаваемый длиной ребра (первый параметр конструктора) и плотностью вещества (второй параметр конструктора), а также класс `Tetrahedron`, представляющий правильный тетраэдр, сделанный из однородного вещества и задаваемый длиной ребра и плотностью.

   В результате следующая функция main():

```
int main() {
    const Body *p, *q, *r;
    Cube a(2, 10), b(5, 0.1);
    Tetrahedron t(6, 2.5);
    p = &a; q = &b; r = &t;
    printf("Volumes: %3.3lf %3.3lf %3.3lf\n",
           p->Volume(), q->Volume(), r->Volume());
    printf("Weights: %3.3lf %3.3lf %3.3lf\n",
           p->Mass(), q->Mass(), r->Mass());
    return 0;
}
```

   должна откомпилироваться без ошибок и предупреждений, отработать и выдать

```
Volumes: 8.000 125.000 25.456
Weights: 80.000 12.500 63.640
```

   Проверьте, что ваша реализация соответствует следующим условиям:

- все поля находятся в закрытой (`private`) части класса, открытыми и защищёнными могут быть только функции-члены;
- директива `friend` не используется;
- для инициализации объектов используются конструкторы;
- никакие методы не изменяют внутренние поля объектов, могут быть только функции, возвращающие значения полей (но не меняющие ничего); как следствие, все методы, кроме конструкторов, помечены как константные;
- данные базового класса нигде не дублируются полями порождённого класса.

3. [10, №10.12] Опишите абстрактный класс `Prism`, представляющий прямоугольную призму с известной высотой, но неизвестной формой основания. Предполагается, что все величины имеют тип `double`. Предусмотрите чисто виртуальную функцию `Square()`, возвращающую площадь основания призмы, и простую функцию `Volume()`, вычисляющую объём как произведение высоты на площадь основания.

   Унаследуйте от класса `Prism` класс Box, представляющий прямоугольный параллелепипед с квадратом в основании, задаваемый длиной бокового ребра (первый параметр конструктора) и длиной стороны основания (второй параметр). Унаследуйте от класса `Box` класс `Cube`, представляющий куб, задаваемый длиной ребра (единственный параметр конструктора). В результате следующая функция `main()`:

```
int main() {
    const Prism *p, *q, *r;
    Box a(0.5, 2), b(5, 0.2);
    Cube c(0.5);
    p = &a; q = &b; r = &c;
    printf("Squares: %3.3lf %3.3lf %3.3lf\n",
           p->Square(), q->Square(), r->Square());
    printf("Volumes: %3.3lf %3.3lf %3.3lf\n",
           p->Volume(), q->Volume(), r->Volume());
    return 0;
}
```

   должна откомпилироваться без ошибок и предупреждений, отработать и выдать

```
Squares: 4.000 0.040 0.250
Volumes: 2.000 0.200 0.125
```

   Проверьте выполнение условий, перечисленных в конце предыдущей задачи.

4. [10, №10.17, 9.11] *Перестановкой* из N элементов называется некий упорядоченный набор, составленный из чисел $1, 2, \ldots, N$, в котором каждое число встречается ровно один раз. Так, перестановок из двух элементов существует две ({1,2} и {2,1}), перестановок из трёх элементов — шесть ({1,2,3}, {1,3,2}, {2,1,3}, {2,3,1}, {3,1,2}, {3,2,1}), а в общем случае существует $N!$ перестановок из N элементов.

   Реализуйте автомат, который при его создании получает число N, а на каждом шаге выдаёт очередную перестановку из N элементов. После исчерпания перестановок, т. е. после того, как функция шага автомата была вызвана $N!$ раз, при последующих вызовах она должна заполнять массив элементов нулями, чтобы показать, что больше перестановок нет. Оформите данный автомат в виде объекта класса `Permutation`, имеющего конструктор для инициализации и задания начальных параметров, деструктор для высвобождения захваченных ресурсов, и один обыкновенный публичный метод, выполняющий шаг автомата. Приватные методы можно вводить без ограничений.

Список литературы

[1] *Вайсфельд М.* Объектно-ориентированное мышление.

[2] *Мейер Б.* Объектно-ориентированное конструирование программных систем.

[3] *Пышкин Е. В.* Основные концепции и механизмы объектно-ориентированного программирования.

[4] *Пол А.* Объектно-ориентированное програмирование на C++.

[5] *Буч Г.* Объектно-ориентированный анализ и проектирование с примерами приложений на C++.

[6] *Андрианова А.А., Исмагилов Л.Н., Мухтарова Т.М.* Объектно-ориентированное програмирование на C++.

[7] *Страуструп Б.* Язык программирования C++.

[8] *Липпман С., Лажойе Ж.* C++ для начинающих.

[9] *Столяров А. В.* Введение в язык Си++.

[10] *Столяров А. В.* Программирование: введение в профессию. Задачи и этюды. — М. МАКС Пресс, 2022. — 156 с.

[11] *Подбельский В. В.* Язык Си++.

[12] http://www.cplusplus.com.

[13] *Пилгрим М.* Погружение в Python 3.

[14] https://www.python.org/doc/.

[15] https://en.wikibooks.org/wiki/Python_Programming.