Семинар #2: Инкапсуляция. Домашнее задание.

Класс Circle

Допустим, что мы хотим создать программу, которая будет работать с окружностями (это может быть игра или, например, графический редактор). Для того, чтобы сделать код более понятным и удобным в использовании, мы решили создать класс окружности. Кроме того, мы решили использовать уже ранее написанный класс точки в 2D пространстве (файлы `point.h` и `point.cpp`). Создайте класс окружности, который будет включать следующие методы:

- Конструктор `Circle(const Point& acenter, float aradius)`, который будет задавать поля `center` и `radius` соответстующими значениями.
- Конструктор по умолчанию `Circle()` - задаются значения, соответствующие единичной окружности с центром в начале координат.
- Конструктор копирования `Circle(const Circle& circle)`
- Сеттеры и геттеры, для полей `center` и `radius`. Поле `radius` нельзя задать отрицательным числом. При попытке задания его отрицательным числом оно должно устанавливаться в значение `0`.
- Метод `float getArea() const`, который будет возвращать площадь поверхности круга.
- Метод `float getDistance(const Point& p) const`, который будет возвращать расстояние от точки `p`, до ближайшей точки окружности.
- Метод `bool isColliding(const Circle& c) const`, который будет возвращать `true`, если круг пересекается с кругом `c`.
- Метод `void move(const Point& p)`, который будет перемещать кружок на вектор `p`.

Весь начальный код содержится в папке `0circle`. При компиляции нужно указывать все `.cpp` файлы, которые вы хотите скомпилировать:

```
g++ main.cpp point.cpp
```

- Создайте файлы `circle.h` и `circle.cpp` и перенесите реализацию класса окружности из файла `main.cpp` в эти файлы.

Класс Number (большое число)

Стандартные целочисленные типы данных, такие как `int` имеют фиксированный небольшой размер. Соответственно значения, которые можно хранить в переменных этих типов ограничены. Типичное максимальное значение `char` равно $2^7 - 1 = 127$, тип `int` обычно ограничен $2^{31} - 1 = 2147483647$ и даже тип `unsigned long long` имеет ограничение в $2^{64} - 1 = 1.8 * 10^{19}$. Хранить действительно большие числа в этих типах невозможно. В этом задании нужно сделать класс, с помощью которого будет удобно складывать и умножать большие целые положительные числа. Начальный код этого класса содержится в `1number/number.cpp`. Изучите этот код.

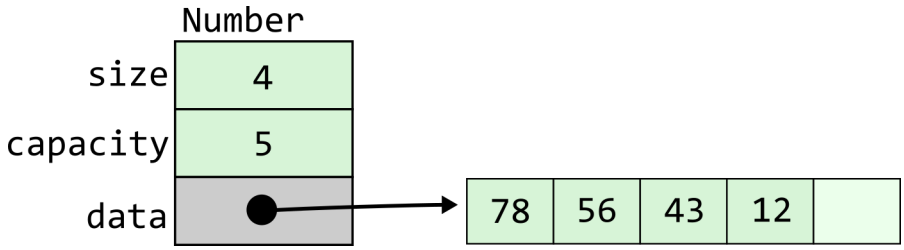

Рис. 1: Представление числа 12345678 в памяти с помощью нашего класса Number

Задания:

- **Конструктор по умолчанию:** Напишите конструктор по умолчанию `Number()`, который будет создавать число равное нулю.
- **Конструктор копирования:** Напишите конструктор копирования `Number(const Number& n)`.
- **Конструктор из строки:** Напишите конструктор `Number(const char* str)`, который будет создавать большое число на основе строки. Предполагаем, что на вход конструктору всегда идёт корректная строка. Например, число из примера можно будет создать так:

  ```
  Number a = Number("12345678");
  ```

- **Присваивание:** Напишите оператор присваивания `Number& operator=(const Number& right)`.
- **Сложение:** Напишите и протестируйте операторы сложения `operator+` и оператор присваивания сложения `operator+=`. Реализовывать оба этих оператора с нуля необязательно. Ведь, если написан один из этих операторов, то очень просто написать другой.
- **Числа Фибоначчи:** Числа Фибоначчи задаются следующим образом:

  $$F_0 = 0$$
  $$F_1 = 1$$
  $$F_n = F_{n-1} + F_{n-2}$$

  Используйте класс `Number`, чтобы вычислить F_{1000}. Правильный ответ:

  ```
  F(1000) = 43466557686937456435688527675040625802564660517371780402481729089536555417949051890
  403879840079255169295922593080322634775209689623239873322471161642996440906533187938298969649928516003704476137795166849228875
  ```

- **Четность:** Напишите метод `bool isEven() const`, который будет проверять является ли наше число чётным и, если это верно, возвращает `true`, в ином случае возвращает `false`.
- **Произведение:** Напишите метод `Number operator*(const Number& right) const` - оператор умножения одного числа `Number` на другое. Протестируйте вашу функцию на различных примерах (умножение большого числа на большое, умножение большого числа на небольшое (< 100) или на ноль, умножение двух небольших чисел и т. д.).

- **Факториал:** Используйте написанный оператор для вычисления факториала от 1000.
  Правильный ответ:

```
1000! = 40238726007709377354370243392300398571937486421071463254379991042993851239862902059
2044208486969404800479988610197196058631666872994808558901323829669944590997424504087073759
9188236277271887325197795059509952761208749754624970436014182780946464962910563938874378864
8733711918104582578364784997701247663288983595573543251318532395846307555740911426241747434
9347553428646576611667797396668820291207379143853719588249808126867838374559731746136085379
5345242215865932019280908782973084313928444032812315586110369768013573042161687476096758713
4831202547858932076716913244842623613141250878020800026168315102734182797770478463586817016
4365024153691398281264810213092761244896359928705114964975419909342221566832572080821333186
1168115536158365469840467089756029009505376164758477284218896796462449451607653534081989013
8544248798495995331910172335555660213945039973628075013783761530712776192684903435262520001
5888535147331611702103968175921510907788019393178114194545257223865541461062892187960223838
9714760885062768629671466746975629112340824392081601537808898939645182632436716167621791689
0977991190375403127462228998800519544441428201218736174599264295658174662830295557029902432
4153181617210465832036786906117260158783520751516284225540265170483304226143974286933061690
8979684825901254583271682264580665267699586526822728070757813918581788896522081643483448259
9326604336766017699961283186078838615027946595513115655203609398818061213855860030143569452
7224206344631797460594682573103790084024432438465657245014402821885252470935190620929023136
4932734975655139587205596542287497740114133469627154228458623773875382304838656889764619273
8381490014076731044664025989949022222176590433990188601856652648506179970235619389701786004
0811889729918311021171229845901641921068884387121855646124960798722908519296819372388642614
8396573822911231250241866493531439701374285319266498753372189406942814341185201580141233448
2801505139969429015348307764456909907315243327828826986460278986432113908350621709500259738
9863554277196742822248757586765752344220207573630569498825087968928162753848863396909959826
2809561214509948717012445164612603790293091208890869420285106401821543994571568059418727489
9809425474217358240106367740459574178516082923013535808184009699637252423056085590370062427
1243416909004153690105933983835777939410970027753472000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000
```

- **Числа-градины:** Возьмём некоторое число n и будем последовательно применять к нему следующую функцию:

$$f(n) = \begin{cases} n/2, & \text{если n - четное} \\ 3n+1, & \text{если n - нечетное} \end{cases}$$

  В результате получится некоторая последовательность. Например, при $n = 7$ получится:

```
7 -> 22 -> 11 -> 34 -> 17 -> 52 -> 26 -> 13 -> 40 -> 20 -> 10 -> 5 -> 16 -> 8 -> 4 -> 2 -> 1
```

  Последовательность доходит до 1. Вам нужно написать функцию, которая будет по начальному числу находить длину такой последовательности (`steps`) и максимальное число в этой последовательности(`max`). Например, для числа 7, максимальное число в последовательности будет равно 52, а длина последовательности – 16. Напишите программу, которая будет по начальному числу находить длину последовательности и максимальный элемент в ней.

  Тесты для проверки:

```
n = 7                    steps = 16;      max = 52
n = 256                  steps = 8;       max = 256
n = 1117065              steps = 527;     max = 2974984576
n = 4761963248413673697  steps = 2337;    max = 99269277123749507446 48

n = 9056079265697294758243978560897246578962897458726405628465872177 1
steps = 1630;
max = 77365802164374936079217113721415149485124440399354098083808056452 0
```

  Для решения этой задачи нужно написать оператор сравнения и метод деления на 2 (оператор целочисленного деления можно не писать).

- **Раздельная компиляция:** Перенесите объявление класса `Number` в файл `number.h`, а определение методов в файл `number.cpp`. Раздельно скомпилируйте эту программу.