Классы

Классы – это классно.

Сначала у нас в программировании появились переменные – некоторые простые числовые объекты, которые можно складывать, вычитать и так далее…

Потом у нас появились функции – некоторые инструкции компьютеру, что нужно сделать. У функций были два типа переменных – аргументы (это те, которые передаются функции в качестве параметра), и локальные переменные (которые создаются прямо в функции). Пример:

```
int factorial(int n) {
    int result = 1;
    while (n > 0) {
        result = result * n;
        n = n - 1;
    }
    return result;
}
```

В этом примере переменная n является аргументом, а переменная `result` – локальной переменной.

В этом листочке вам для изучения предлагается ещё одна сущность, которая называется **класс**. Эта штука представляет собой произвольный объект, у которого имеется, во-первых, некоторое внутреннее **состояние**. Например, у объекта «точка» внутренним состоянием являются её координаты. У человечка в компьютерной игре может быть жизнь и мана. У паспорта – имя и фамилия его владельца. У компьютера – запущенные программы, например. И, во вторых, у этого объекта есть некоторый набор **функций**, которые могут с этим состоянием работать. Например, у компьютера должны быть функции «вернуть список запущенных программ», «запустить новую программу», «закрыть эту программу». У человечка в игре – «в тебя попали», «колдовать такое-то заклинание».

Попробуем для начала написать класс «точка на плоскости». Итак, какие у точки внутренние переменные? Это две координаты типа double. Так и напишем:

```
class Point {
  private:
    double x_;
    double y_;
};
```

Что здесь за что отвечает? Мы описываем класс Point, для чего пишем сначала class Point. Затем описываем содержимое этого класса – ставим {, } и не забываем ;. Получается пустой класс Point:

```
class Point {
};
```

Теперь части содержимого: перед переменными, которые отвечают за внутреннее состояние объекта, пишем `private` и затем перечисляем эти «внутренние переменные» - в данном случае их роль играют double x_ и double y_;

Пока что в нашем классе нет функций, которые могли бы работать с внутренними данными. Давайте их добавим.

Во-первых, будет полезным сделать функции, которые просто будут возвращать координаты x и y точки.

```
class Point {
  private:
    double x_;
    double y_;

  public:
    double GetX() {
        return x_;
    }

    double GetY() {
        return y_;
    }
};
```

Итак, смотрим, что изменилось. Добавилось слово public. Оно пишется перед функциями класса (они не private, не собственные, а общие). А ещё мы написали сами функции. Заметим, что, например, функция GetX работает только с переменной класса, а именно с x_. У неё нет ни локальных переменных, ни аргументов.

Попытаемся использовать наш класс.

```
int main() {
    Point p; //Таким образом создаётся обьект p класса Point. Прямо как int number;
    cout << p.GetX();   //А здесь мы вызываем функцию GetX.
    return 0;
}
```

Итак, точка у нас создаётся точно так же, как создавалось бы число – пишем сначала тип, потом имя. В результате одновременно с p у нас создались внутренние переменные p.x_ и p.y_. Теперь можно вызывать методы нашего класса – пишем p.GetX().

Мы столкнулись с одной маленькой проблемой. А именно: пока что мы предусмотрели никакого способа назначить точке координаты, и поэтому p.GetX() нам пока возвращает какую-то чушь. Конечно, можно сделать функцию `SetX(double new_x)`, которая будет присваивать переменной x_ значение `new_x`. Но в языке C++ для этого есть специальный механизм, который называется конструктором. **Конструктор** – это функция в классе, которая называется точно так же, как и сам класс, ничего не может возвращать, и которая вызывается в момент создания объекта этого класса. То есть когда мы пишем Point p, то у нас сначала создаются переменные p.x_ и p.y_, а затем вызывается конструктор. Пример:

```
class Point {
... //Здесь тот же текст, что и был раньше
    Point(double init_x, double init_y) { //Конструктор
        x_ = init_x;
        y_ = init_y;
    }
};
```

```
int main() {
    Point P(2.5, 3.5); //Чтобы вызвать конструктор с аргументами, эти аргументы пишутся вот так вот
    cout << p.GetX();   //И снова мы вызываем функцию GetX.
    return 0;
}
```

В этом случае нам выведется 2.5, а не какое-то случайное число.

Кстати, конструктор можно написать не один. Пример:

```
class Point {
  private:
    double x_;
    double y_;

  public:
    double GetX() {
        return x_;
    }

    double GetY() {
        return y_;
    }

    Point(double init_x, double init_y) { //Конструктор №1
        x_ = init_x;
        y_ = init_y;
    }
    Point() { //Конструктор №2
        x_ = 0;
        y_ = 0;
    }
};

int main() {
    Point P1(2.5, 3.5);
    Point P2;
    return 0;
}
```

Как мы видим, при создании P1 мы передавали какие-то параметры. Компилятор **автоматически** найдёт подходящий конструктор – в данном случае он ищет конструктор, у которого были бы два числа в качестве аргументов. А вот при создании P2 вызовется второй конструктор, поскольку никаких параметров передано не было.

Теперь напишем ещё один класс Segment – отрезок. У отрезка есть две переменных состояния – точка начала и конца. А ещё пусть он, например, умеет возвращать свою длину. Вот этот класс:

```
class Segment {
  private: //Внутренние переменные
    Point A_;
    Point B_;

  public:
    Segment(Point A, Point B) { //Конструктор №1
        A_ = A;
        B_ = B;
    }

    double GetLength() {
        return sqrt((A_.GetX() - B_.GetX()) * (A_.GetX() - B_.GetX()) +
                    (A_.GetY() - B_.GetY()) * (A_.GetY() - B_.GetY()));
};
```

Как вы видите, мы с лёгкостью воспользовались здесь уже написанным классом Point. После этого можно написать вот такой вот main:

```
int main() {
    Point A(1.0, 2.0);
    Point B(3.0, 4.0);
    Segment a(A, B);
    cout << a.GetLength();
    return 0;
}
```

Задача A. Генератор случайных чисел

Реализуйте класс генератора случайных чисел Random. Он должен иметь два метода – Get, который возвращает очередное целое случайное число, метод GetDouble, который возвращает случайное число от 0 до 1, и два конструктора – с инициализатором по времени и с инициализатором по заданному числу.
Примечание. Эта задача была написана и разобрана на уроке. Если вы это пропустили, то ничего страшного – просто пользуйтесь уже написанным кодом.

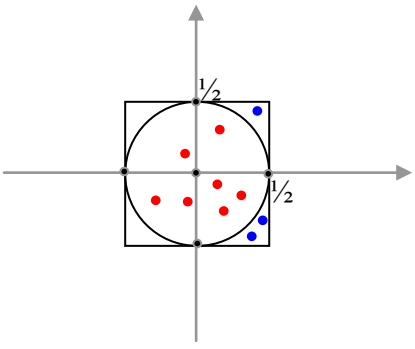
Решение:

```
#include <iostream>
#include <time.h>
#include <limits>

using namespace std;

class Random {
public:
    unsigned int Get() {
        previous_random_number_ = a_ * previous_random_number_ + c_;
        return previous_random_number_;
    }
```

```
    double GetDouble() {
        double random = Get();
        random = random / std::numeric_limits<unsigned int>::max();
        return random;
    }

    Random() {
        a_ = 5;
        c_ = 69069;
        previous_random_number_ = time(NULL);
    }

    Random(int seed) {
        a_ = 5;
        c_ = 69069;
        previous_random_number_ = seed;
    }

private:

    unsigned int previous_random_number_;
    unsigned int a_;
    unsigned int c_;

};

int main() {
    Random r1;
    cout << r.Get() << endl;
    Random r2(10);
    cout << r2.Get() << endl;
    return 0;
}
```

Задача B. Число π/4

Необходимо воспользоваться предыдущим классом, дабы вычислить число π методом Монте-Карло. Метод заключается в следующем: как известно, круг радиуса ½ имеет площадь π/4. Рассмотрим круг радиуса ½ с центром в 0 и опишем вокруг него квадрат со стороной 1. Тогда площадь квадрата – 1. Начнём генерировать случайным образом точки в этом квадрате, вызывая функцию GetDouble (случайная точка будет иметь координаты (generator.GetDouble() - 0.5, generator.GetDouble() - 0.5)). Вычислим количество точек, которые попадут в круг – $N_{попали}$. Также пусть всего было сгенерировано N точек. Тогда площадь круга (и, соответственно, искомое число), будет равна $N_{попали}/N$. В приведённом примере для N = 10 мы получили $N_{попали}$ = 7, итого получили, что площадь равна примерно 0,7.



Напишите программу, которая считывает число N с клавиатуры и выдаёт в качестве ответа $N_{попали}/N$.

Задача C. Точка

Реализуйте класс Point, который знает свои 2 координаты и умеет вычислять расстояние до точки (0, 0). Вот примерный вид этого класса:

```
class Point {
  public:
    Point(double init_x, double init_y) {
        // Здесь какой-то код
    }

    double GetDistanceTo0() {
        // Здесь какой-то код
    }

  private:
    double x_;
    double y_;
};
```

Также напишите код, который считает расстояние от точки до 0. Примерно такой:

```
int main() {
    double x;
    double y;
    cin >> x;
    cin >> y;
    Point p(x, y);
    cout << p.GetDistanceTo0() << endl;
    return 0;
}
```

Входные данные	Выходные
0.5 0.5	0.707

После этого в классе Point допишите также метод
double GetDistanceToPoint(Point other),
который бы вычислял расстояние до другой точки.

Задача D. В точку.

Добавьте в класс Random функцию GetPoint, которая бы возвращала случайную точку. Перепишите задачу B с помощью этой функции.

Задача E. Центр тяжести

Выведите координаты центра тяжести данного множества точек.

Сохраните исходные данные в массиве объектов Point. Программа получает на вход набор точек на плоскости. Сначала задано количество точек *n*, затем идет последовательность из *n* строк, каждая из которых содержит два числа: координаты точки. Величина *n* не превосходит 100, все исходные координаты – целые числа, не превосходящие 103.

Задача F. Диаметр множества

Выведите диаметр данного множества точек – максимальное расстояние между двумя точками из множества. В реализации программы должна быть функция вычисления расстояние между двумя точками.

Программа получает на вход набор точек на плоскости. Сначала задано количество точек n, затем идет последовательность из n строк, каждая из которых содержит два числа: координаты точки. Величина n не превосходит 100, все исходные координаты – double, не превосходящие 10^3.

Задача G. Влюблённость.

У всех людей на планете Планета есть два параметра – красота и ум (типа double). Про любого мужчину каждой женщине (да и мужчине) сразу ясно, насколько он умён или красив. Женщина влюбляется в мужчину, если он красивее её, а она умнее его (да-да, именно так). Напишите классы Man и Woman, которые хранят свои параметры, а также метод

```
bool Woman.InLove(Man man),
```

который возвращает true, если женщина влюбилась в мужчину. Кстати, каждый раз, когда женщина влюбляется, она становится на 5 процентов глупее и на 5 процентов красивее.

Теперь задача: женщина с параметрами woman_beauty и woman_cleverness идёт по улице, и встречает N мужчин с параметрами (MB_1, MC_1), …, (MB_N, MC_N) (первый параметр – всегда красота, второй параметр – всегда ум). Для каждого мужчины необходимо дать ответ, влюбится ли она в него или нет.

Входные данные:

На первой строке указаны параметры woman_beauty, woman_cleverness и N. Затем идёт N сточек, каждая из которых – параметры красоты и ума для каждого из встреченных мужчин.

Входные данные	Выходные
100 100 5	
1 1	NO
2 2	NO
3 3	NO
4 4	NO
5 5	NO

10.0 10.0 9	
9.4 11.1	NO
9.4 9.3	NO
12.3 10.1	NO
12.4 9.7	YES
9.4 11.1	NO
9.4 9.3	NO
12.3 10.1	NO
12.4 9.7	NO
12.4 9.3	YES

Задача H. Рациональные числа.

Реализуйте класс рационального числа Rational (то есть отношение двух целых чисел, знаменатель всегда больше 0). У него должны быть методы:

```
void Add(Rational other) // (прибавить к себе другое число)
void Sub(Rational other) // (вычесть другое рациональное число)
void Mult(Rational other)// (умножить на другое рациональное число)
void Div(Rational other) // (разделить)
double GetDecimal()      // вернуть приблизительную десятичную дробь
void Print()             // (напечатать на экране в виде a/b)
```

И конструкторы

```
Rational(int nominator, int denominator)
Rational(double number)
Rational(Rational other)
```

Всякие сокращения и приведения к нормальному виду реализовывать необязательно, дробь может быть несокращённой.

Если знаменатель дроби тем или иным способом стал равен 0, то числитель тоже необходимо занулить.

Напишите программу, которая считывает с клавиатуры два дробных числа типа double и выводит на экран их отношение в виде дроби.

Входные данные	Выходные
0.5 0.5	50/50
0.3 1.8	30/180

Задача I. (*) Сокращённая дробь.

Реализуйте класс Rational, в котором дробь будет всегда сокращённой.

Задача J. Число e.

Напишите программу, которая, используя класс Rational, вычисляла бы приблизительно число *e* с помощью формулы $e \approx (1 + 1/n)^n$ для $n = 9$. Попробуйте вычислить для $n = 15$. Почему получилось нечто странное? Измените тип внутренних полей числителя и знаменателя с int на double. Что-нибудь изменилось?

Задача K. Сумма третей.

Напишите программу, которая, используя класс Rational, вычисляла бы выражение 1/3 + 1/9 + … + 1/3^18 – 1.

Задача L. Средний балл по предметам

Заданы сначала количество учащихся n, затем n строк, каждая из которых содержит фамилию, имя и три числа (средние оценки по трем предметам: математике, физике, информатике). Данные в строке разделены одним пробелом. Оценки принимают значение от 1 до 5. Количество учеников не больше чем 100. Выведите три действительных числа: средний балл всех учащихся по математике, по физике, по информатике. *В программе должны обязательно присутствовать отдельная функция, которая выводит средний балл массива учеников.*

Задача M. Отличники и хорошисты

Заданы сначала количество учащихся n, затем n строк, каждая из которых содержит фамилию, имя и три числа (оценки по трем предметам: математике, физике, информатике). Данные в строке разделены одним пробелом. Оценки принимают значение от 1 до 5. Необходимо вывести пары фамилия-имя по одной на строке, разделяя фамилию и имя одним пробелом, для тех учеников, у которых все средние оценки больше либо равны 4. Выводить оценки не нужно. Порядок вывода должен быть таким же, как в исходных данных. *В программе должны обязательно присутствовать отдельная функция, которая выводит отличников и хорошистов по массиву учеников.*

Задача N. Трое лучших

Заданы сначала количество учащихся n, затем n строк, каждая из которых содержит фамилию, имя и три числа (оценки по трем предметам: математике, физике, информатике). Данные в строке разделены одним пробелом. Оценки принимают значение от 1 до 5. Необходимо вывести пары фамилия-имя по одной на строке, разделяя фамилию и имя одним пробелом, троих учеников с самым лучшим средним баллом. Выводить оценки не нужно. Порядок вывода должен быть таким же, как в исходных данных. *В программе должны обязательно присутствовать отдельная функция, которая выводит троих лучших из массива учеников.*

Задача O. Максимальный треугольник

Среди исходных точек найдите три, образующие треугольник максимальной площади. Выведите данную площадь. Программа получает на вход набор точек на плоскости. Сначала задано количество точек n ($2<n<101$), затем идет последовательность из *n* строк, каждая из которых содержит два числа: координаты точки. Все исходные координаты – целые числа, не превосходящие 103.

В данной задаче требуется создать класс, описывающий треугольник на основе трех точек и умеющий вычислять свою площадь, а также вспомнить формулу Герона из листков прошлого года.

Задача P. Физика

Маленький твёрдый шарик бросили вверх с земли с начальной скоростью v_0. Напишите класс, содержащий два метод – double GetHeight(), который будет возвращать его текущую высоту, и void Tick(double s), которая заставляет шарик двигаться в течении s секунд. g считать равным 10.

Задача Q. Физика№2

Возьмите предыдущую задачку и добавьте к ней ещё два измерения. После чего добавьте ещё один метод void Push(double vdx, double vdy, double vdh), который изменяет скорость шарика на (vdx, vdy, vdh) по каждому из направлений.

Задача R. Стек

Помните, как мы писали стек? Нет? Бегом в \\Sch25-srv\prog\Practice\Programming\M2009\Архив\8m-stack.doc!

А теперь мы напишем его ещё раз. Но теперь уже у нас должен получиться класс Stack, у которого есть методы

void Push(int n)

Добавить в стек число n.

void Pop()

Удалить из стека последний элемент.

int Top()

Последний элемент стека.

int Size()

Количество элементов в стеке.

Clear()

Очистить стек.

Внутри этот стек реализован с помощью массива заранее заданной длины.

Затем напишите такую же управляющую программу, как и в задаче «D» листочка «Структуры данных».

Пример протокола работы программы

Ввод	Вывод
push 2 push 3 push 5 top size pop size push 7 pop clear size exit	ok ok ok 5 3 5 2 ok 7 ok 0 bye

Задача S. Очередь №1

Реализуйте класс Queue (очередь, смотри задачу F из уже указанного листка) с помощью двух Stack.

Задача T. Очередь №2

Реализуйте класс Queue (см. предыдущую задачу) на базе 1 массива.

Задача U. Школьная очередь

За булочками в школьный буфет за переменную выстраивается очередь. Известно, что у каждого школьника есть параметр наглость (типа double). Когда школьник приходит в столовую, то вместо того, чтобы встать в конец очереди, он пытается пробиться вперёд, и у него это получается, пока перед ним не окажется школьник столь же (или более) наглый, как и он сам. При этом тот, кто первый в очереди, его точно вперёд не пустит. Чтобы продвинуться вперёд на одно место, школьнику требуется 1 секунда.

Столовая не обслуживает школьников, если прозвенел звонок. Известны время обслуживания одного школьника в секундах T, длительность в секундах перемены L, количество школьников N, которые пытаются за эту перемену купить булочку, а также наглость каждого из школьников и время его прихода (в секундах от начала перемены) в столовую в порядке их прихода в столовую. Сколько человек в такой очереди будут обслужены за перемену?

Входные данные:

Первая строка – числа T, L, N. Затем следует N строчек, на каждой из которых написана (в порядке входа в столовую) наглость очередного школьника и время его прихода в столовую.

Выходные данные:

Одно целое число – количество школьников, которые будут обслужены за эту перемену.

Входные данные	Выходные
8 100 5 1 1 1 2 1 3 1 4 1 5	5
21 100 5 1 1 1 2 1 3 1 4 1 5	4

Задача V. Вася в очереди

Вася хочет понять, сможет ли он купить булочку. Помогите ему!

Данные – точно такие же, как и в предыдущей задаче, но в первой строке также указано, каким по счёту пришёл Вася. В качестве ответа нужно выдать YES или NO

Входные данные	Выходные
8 100 5 5 1 1 1 2 1 3 1 4 1 5	YES

21 100 5 5 1 0 1 2 1 3 1 4 1 5	NO
3 9 4 4 1 0 1 2 1 3 10 5	YES
3 8 4 4 1 1 1 2 1 3 10 5	NO