Homework 8

**90 градусов**

Вам заданы несколько строчек текста (сначала количество строчек, потом сами строчки). Вам надо перевернуть строчки на 90 градусов как в примере.

|  |  |
| --- | --- |
| 3  Hello World  abcd  a b c d | Haa  eb  lcb  ld  o c  W d  o  r  l  d |
| 4  ab   x  cdе | a  c  bx d     е |

**Подвох:**Наивный подход ко вводу данных приведет к ошибке.

int n;

ctd::cin >> n;

for(int i = 0; i < n; ++i)

{

    std::string currentLine;

    std::getline(std::cin, currentLine);

    ….

}

Это сработает не так как вы ожидаете по следующей причине. Когда вы вводите число n, указатель ввода все еще стоит на строчке с числом (на символе перехода строки). Следовательно следующий getline прочитает остаток этой строки (то есть пустую строку). Чтобы избежать этого прочтите всю строку с числом до конца.

int n;

std::cin >> n;

std::string restOfFirstLine;

std::getline(std::cin, restOfFirstLine);

for(int i = 0; i < n; ++i)

{

    std::string currentLine;

    std::getline(std::cin, currentLine);

    ….

}

**Змейка**

Задано число N. Заполните квадратную матрицу NxN числами от 1 до N2 “змейкой”, как показано в примере.

|  |  |
| --- | --- |
| 4 | 1 8  9 16  2 7 10 15  3 6 11 14  4 5 12 13 |
| 3 | 1 6 7  2 5 8  3 4 9 |

**Степень двойки**

По заданному положительному числу n < 264 определить, является ли оно степенью двойки. Решение должно иметь сложность O(1).

|  |  |
| --- | --- |
| 1 | Yes |
| 17 | No |
| 1024 | Yes |

Внимание: 64-битные числа нужно хранить в типе unsigned long long

**IP адрес**

IP-адрес - это четверка a.b.c.d, где a, b, c, d целые числа из интервала [0,255]. Требуется написать функции encode и decode:

unsigned int encode(const std::string& ipAddress);

std::string decode(unsigned int code);

Первая функция превращает ай-пи адрес в четырехбайтовое число. Функция decode выполняет обратный процесс. Неважно, как написаны функции. Главное, чтобы encode(decode(x)) было равно х для любого беззнакового целого х; и чтобы  decode(encode(strIp)) было равно strIp для любой строки, представляющей собой айпи адрес.

std::cout << decode(encode(“125.0.0.4”)) ; //prints 125.0.0.4

std::cout << encode(decode(19)) ; //prints 19

**Пропускная способность**

В контейнер опущено 10 шлангов. Известны пропускные способности этих шлангов, т.е. сколько кубических сантиметров воды вливается из данного шланга в контейнер. Конфигурация шлангов (т.е. которые из них закрыты, а какие открыты) задается с помощью единственного числа 0 <= C < 210. i-ый бит числа C указывает, открыт ли шланг с номером i. Например, если C = 17, т.е. C = 0000010001, то открыты краны с номерами 5 и 10, а остальные закрыты. При C = 0 все краны закрыты. При C = 210-1, все краны открыты.

Требуется определить суммарную пропускную способность открытых кранов.

**Вход:**

На вход программы поступают 10 целых чисел, пропускные способности кранов, начиная с первого. Далее следует число 0<=C<=1023.

**Выход:**

Вывести суммарную пропускную способность всех открытых кранов

|  |  |
| --- | --- |
| 10 20 30 40 50 60 70 80 90 100  15 | 340 |
| 10 20 30 40 50 60 70 80 90 100  17 | 160 |
| 10 20 30 40 50 60 70 80 90 100  0 | 0 |
| 10 20 30 40 50 60 70 80 90 100  1023 | 550 |

**Куча камней (Тимус 1005)**

http://acm.timus.ru/problem.aspx?space=1&num=1005

**Два Танка**

Друг перед другом стоят два танка. Каждый из них имеет некоторое количество “здоровья”, а также характеристики атаки и защиты. Танки стреляют друг в друга и ваша задача после каждого выстрела напечатать информацию о танках. Стрельба заканчивается, когда один из танков уничтожится (“здоровье будет <= 0”).

**Здоровье:**Здоровье танка выражается целым числом. Изначально здоровье должно быть строго положительным.

**Атака:**Атака танка выражается действительным положительным числом. Это есть сила удара танка при полном здоровье. Пусть здоровье танка h, а начальное здоровье - h0. Тогда сила удара танка с атакой a равна a\*h/h0. То есть раненные танки слабее.

**Защита:**Защита танка выражается действительным числом из диапазона [0,1]. Защита 1 означает, что танк бессмертный, то есть ему невозможно нанести урон. В общем случае, при ударе силы x танку с защитой d наносится x(1-d) урона.

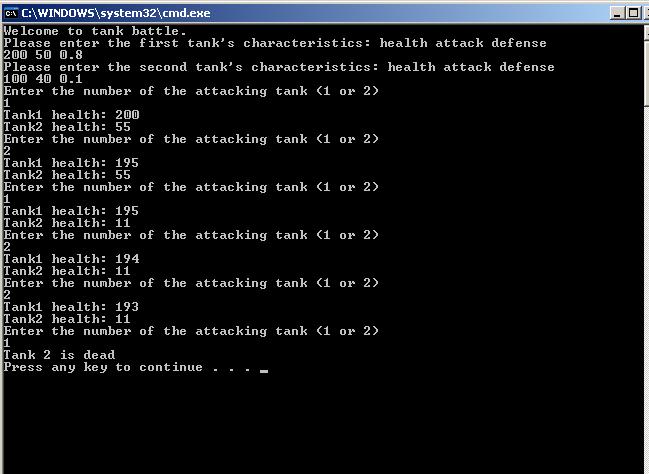
**Удар:**Пусть танк с начальным здоровьем H0, текущим здоровьем H и атакой А наносит удар по танку со здоровьем h и защитой d. Тогда, состояние стреляющего танка не меняется, а новое здоровье h1 танка  под ударом вычисляется по формуле

**h1 = целая часть(h - A(1-d)H/H0))**

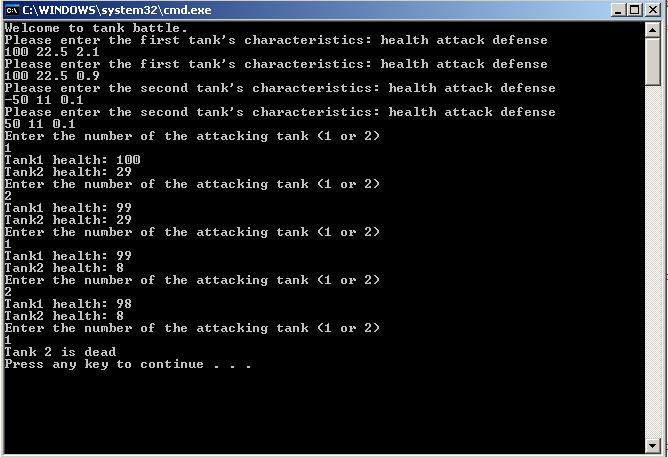
**Вход/Выход:**

Сначала ввести информацию о двух танках, т.е. начальное здоровье, атаку и защиту каждого из танков. Вход должен проверяться на корректность(см.пример2). После этого, пока оба танка живы, последовательно вводить номер стреляющего танка (1 или 2) и после каждого выстрела печатать жизни обоих танков. Как один из танков умрет, напечатать сообщение об этом и завершить программу.

**Пример 1:**



**Пример 2**



**Требования к исходному коду программы.**

Программа должна быть написана с использованием структур и функций. В частности, необходимо иметь структуру Tank описывающую танк, а также следующие функции:

//выводит пригласительное сообщение для ввода информации о танке

//до тех пор, пока не будут введены корректные данные. Испоьзуйте в этой фунцкии цикл do и функцию Validate (см. ниже)

void Input(Tank& tank);

//проверяет данные танка на корректность

bool Validate(const Tank& tank);

//Симулирует удар. Первый аргумент - стреляющий танк (его состояние не меняется, следовательно передаем по ссылке на константу), а второй - обороняемый. Возвращает значение, указывающее, умер ли обороняемый танк.

bool Hit(const Tank& attacker, Tank& defender);

Помимо этих функций могут и скорее всего должны быть и другие фунции.

[Google Диск](https://docs.google.com/)–[Сообщение о нарушении](https://docs.google.com/abuse?id=1MhnUJ7R7nMV_x1Amek1veNpdIKJOsSUnkU2zjLyj_-I)–Интервал автоматического обновления в минутах: 5